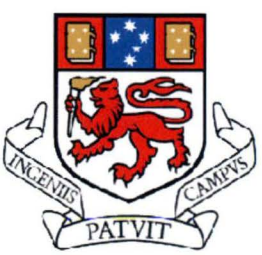
Study of Predictive Models for Emissions in Hydrogen Assisted Dual Fuel Internal Combustion Engine

by

Viet Tien Phuong Nguyen
B.Eng. (Mechatronics Eng., Hons.)

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy



School of Engineering, University of Tasmania
November 2007

Statement of Originality & Authority of Access

This thesis contains no material which has been accepted for a degree or diploma by the University of Tasmania or any other institution, except by way of background information and has been duly acknowledged in this thesis, and to the best of the author's knowledge and belief no material has previously been published or written by another person except where due acknowledgement is made in the text of this thesis.

This thesis may be made available for loan. Copying of any part of this thesis is prohibited for two years from the date this statement was signed; after that time limited copying is permitted in accordance with the Copyright Act 1968.

VIET TIEN PHOONG NGUYEN

School of Engineering,
University of Tasmania,
Hobart, Tasmania, Australia

Abstract

A detailed understanding of the exhaust gas emissions can forecast the state of the engine performance and the other detrimental health effects it can have on the general population. There is no doubt that exhaust gas emissions generated by various internal combustion engines can provide environmental implications. With modern trends in alternative fuels and their mix with the conventional gasoline, is yet another effort to reduce exhaust gas emissions and adhere to strict emissions requirements in automobiles. While a good understanding of the quantitative and qualitative trends are available in the literature, for petrol driven vehicles, little or no published evidence is available for hydrogen vehicles or dual fuel driven vehicles assisted with hydrogen. A good understanding of the near zero emissions and associated conversion technology, using hydrogen as fuel, has been in the domain of few automotive companies around the world. While hydrogen is recognised as a potential fuel of the future, little or no evidence is available in the public domain on the mechanical and electrical conversion technologies and associated emission data for better understanding of this emerging alternative fuel. Conventional engine management systems with their inherent ability to map a particular fuel needed to be modified with dual fuel injection and particular add-on modular tools to accommodate hydrogen injection.

This work is aimed at converting a commercially available Kawasaki Ninja 600cc motorcycle engine to run on both hydrogen and petrol. In this thesis, a rigorous design for conversion to run on hydrogen is designed and built from first principles. The test rig development associated with the calculations for fuel flow rates and associated engine management systems are integral part of this overall systematic design. As part of this investigation, an innovative fuel injection system together with add-on injection system is developed. Using artificial neural networks, predictive models for various mixtures of hydrogen-petrol are developed to estimate emissions for various hydrogen-petrol mixtures. It is argued in this thesis that the accuracy of prediction for emissions can replace expensive gas emissions equipment so that the intelligent mathematical predictive tools can be used as virtual sensors. As part of this investigation a comprehensive range of engine operating conditions is tested using both petrol and hydrogen as fuel for various combinations. The predictive model as virtual sensors has shown that the

predictive capability for emissions is close to ±10% for various combinations of hydrogen-petrol fuel mix. Exhaust emission performance showed significant reduction in oxides of nitrogen and no significant emissions of hydrocarbons, carbon dioxide and carbon monoxide with increasing percentages of hydrogen injection. This work is seen a step towards understanding the intricate hydrogen conversions, development of add-on electronic injection control units to accommodate hydrogen and neural network based predictive models, as virtual sensors, to estimate internal combustion performance.

Acknowledgements

I am deeply indebted to my supervisor, Dr. Vishy Karri. Without his endless technical support, guidance and recommendations throughout the duration of the research, this project would not have completed. My motivation for research was significantly influenced by his enthusiasm for research and knowledge. His encouragement had actually brought me back on track whenever I felt frustrated and stressed.

I would like to to express my grattitude towards the University of Tasmania, the technical and workshop staffs at the School of Engineering, especially Mr. Glenn Mayhew. Generous and invaluable technical support from the staffs has always been greatly appreciated. In addition to that, without the scholarship offered by the University of Tasmania I would not have been able to enter the Ph.D. program.

I would also like to thank all of my friends who have supported and assisted me during my study.

Finally, I would like to show my appreciation to my parents, my younger sister and my relatives. Without their support and emotional care, I would have achieved nothing.

Table of Contents

CHAPTER 1
INTRODUCTION

1.1. AIR POLLUTION AND GLOBAL WARMING PROBLEMS

1.1.1. Motor vehicle growth

Motor vehicles since the first day of their presence have brought a gigantic benefit to human's life and play a significant role in the course of development of civilisation on earth. They provide a flexible mean of transport for people to move between home and work. They allow goods to be delivered to wide distribution networks. Therefore, it is not exaggerating to say that motor vehicle invention is among the most important ones in the history.

However, looking at the use of motor vehicles at a different angle, one can also tell the disadvantages and impacts on other aspects of human life and environment. Environmental and economic costs would be incurred by us when the number of automobile being used increases, which has already happened in the last few decades. Vehicles are considered a major source of urban air pollution and green house gas emissions. Widespread use of vehicles would also increase the dependence on fossil fuel sources and, with the fact that those sources are depleting quickly, it sometimes causes conflict between nations.

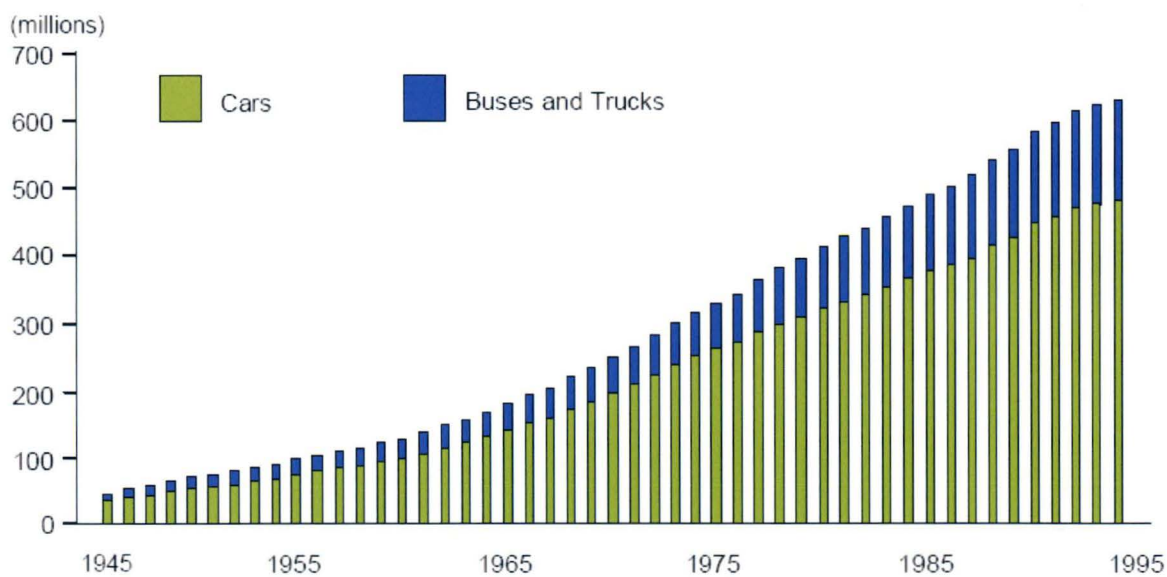


Figure 1.1: *Trends in Global Motor Vehicle Registration, 1945-95* [1]

Since about 1970, the global fleet has been growing at a steady rate of about 16 million vehicles per year (Figure 1.1). Interestingly, this expansion has been accompanied by a similar linear growth in fuel consumption [3]. If the growth of global vehicle fleet continues to rise linearly, it is expected to have about 1 billion vehicles around the world by the year 2025.

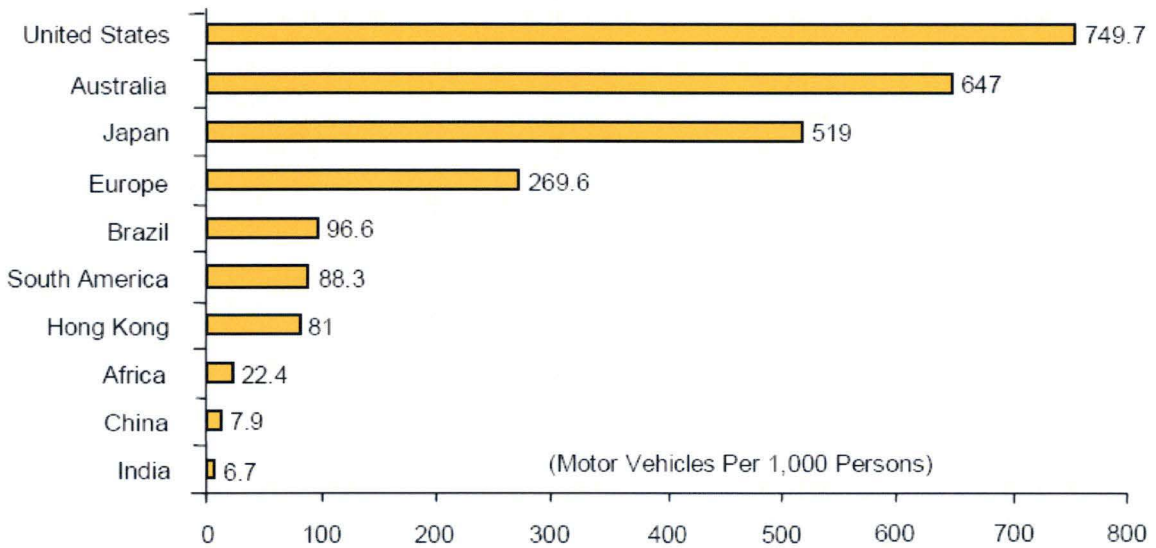


Figure 1.2: *Motor Vehicle Registrations in Selected Countries and Regions, 1994* [2]

Car ownership is higher in rich countries like the US, Australia, Japan, Europe than in other developing countries. For instance, in 1994, there were approximately 7 vehicles registered per 1000 people while there were about 750 vehicles registered per 1000 persons in the US in the same year, which is about 100 times (Figure 1.2).

1.1.2. Vehicle emissions and environment impact

Motor vehicles are the major source of urban air pollution. According to EPA [4], in Melbourne, motor vehicle emissions contribute the following levels of pollutants to the overall air quality:

- 80 per cent of Carbon Monoxide (CO).
- 60 per cent of Nitrogen Oxides (NO_X).
- 40 per cent of Volatile Organic Compounds (Hydrocarbons).
- 30 per cent of Particulate Matter (PM) (resulting in winter smog)

Besides causing impact on the quality of urban air, automobiles exhaust gases also contribute a great portion in elevation of the average global temperature, which is also well known as global warming.

Australia has officially recorded its warmest year on record. Data collected by the Bureau of Meteorology (Figure 1.3) indicates that the nation's annual mean temperature for 2005 was 1.09°C above the standard 1961-90 average, making it the warmest year since reliable, widespread temperature observations became available in 1910. The previous record of +0.84°C was set in 1998. While these temperature departures may seem relatively small, a 1°C increase in mean temperatures is equivalent to many southern Australian towns shifting northward by about 100km. [6]

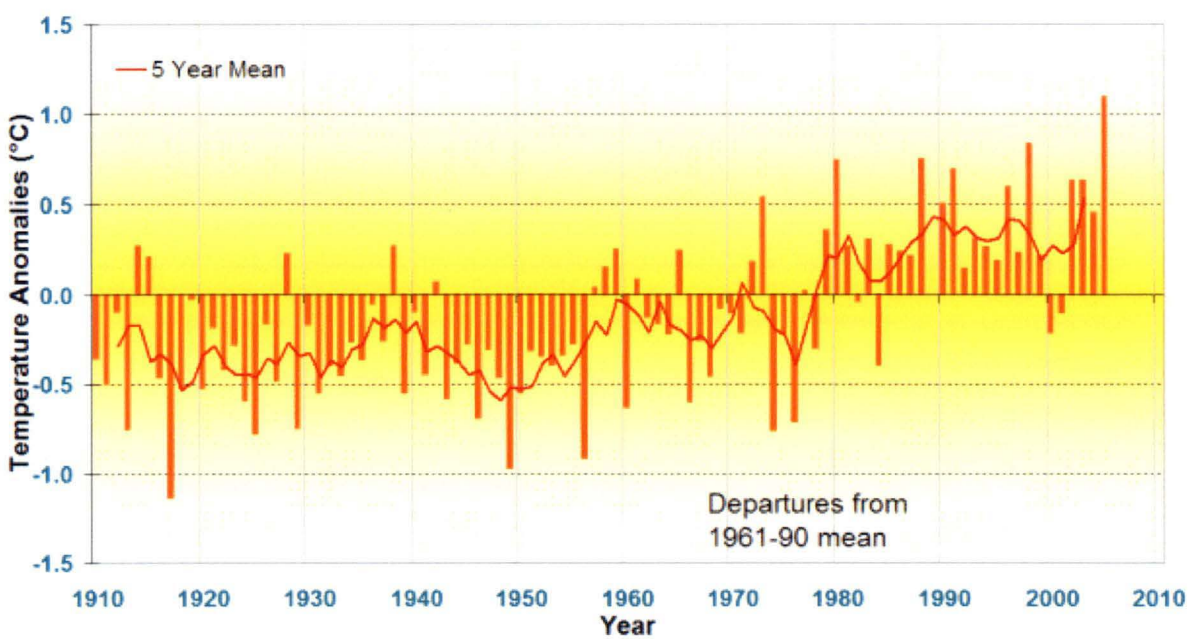


Figure 1.3: *Annual mean temperature anomalies for Australia* [6]

The warm conditions in 2005 were remarkably widespread. All States and Territories, apart from Victoria and Tasmania, recorded 2005 mean temperatures amongst their top two warmest years on record (Figure 1.4). The only region recording a cooler than normal year was a coastal strip of Western Australia extending from Cape Leeuwin to Carnarvon. [6]

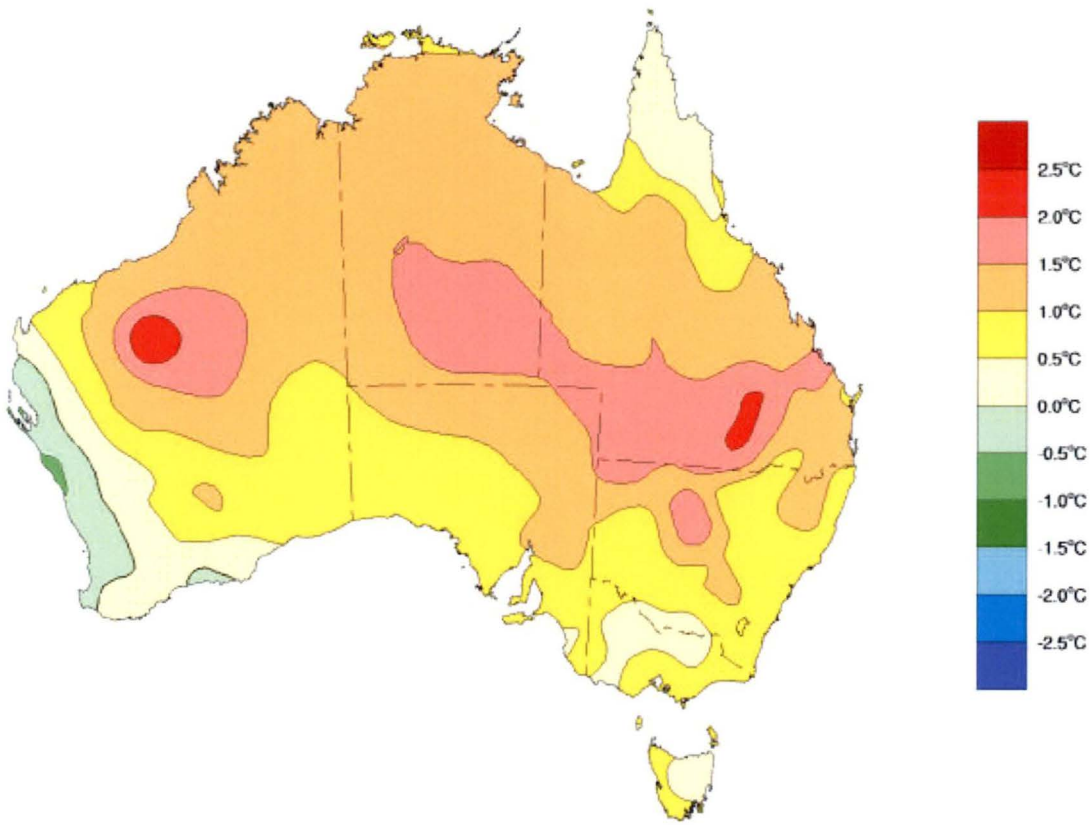


Figure 1.4: *Mean temperature anomalies (^{0}C) for the period from 1/1/2005 to 31/12/2005* [6]

Vehicle emissions contribute to global warming with Carbon Dioxide (CO_2) and Nitrous Oxide (NO_X). CO_2 contributes 64% to the greenhouse effect while the percentage of NO_X is 6% (Figure 1.5) while NO_X is approximately 270 times more powerful than CO_2 at trapping heat in the atmosphere [7].

Gas	Percentage
Carbon Dioxide	64%
Methane	19%
Nitrous Oxide	6%
CFC-12	6%
Other Halocarbons	5%

Figure 1.5: *Share of greenhouse warming due to different greenhouse gases* [5]

1.2. EMISSION IMPACTS ON HUMAN HEALTH

Emission components from motor vehicles that can produce health effects are Carbon Monoxide (CO), Nitrogen Oxides (NO_X), Volatile Organic Compounds (Hydrocarbons), and Sulphur Dioxide (SO_2). In addition to that, by products from automobile exhaust gases such as Ozone (O_3) and particulate matters such as Sulphates (SO_4^-) and Nitrates (NO_3^-) are also the sources that result in human health problems.

1.2.1. Carbon Monoxide (CO)

Carbon Monoxide is an odorless gas and is readily absorbed from the lungs into the blood stream, which then reacts with Haemoglobin molecules in the blood to form Carboxyhaemoglobin. This reduces the Oxygen carrying capacity of blood, which in turn impairs Oxygen release into tissue and adversely affects sensitive organs such as the brain and heart [8].

1.2.2. Nitrogen Dioxide (NO_X)

Nitrogen Dioxide group includes Nitric Oxide (NO) and Nitrogen Dioxide (NO_2). When escapes to the ambient air, Nitric Oxide (NO) is oxidized to Nitrogen Dioxide (NO_2). Nitrogen

Dioxide (NO_2) contributes both to morbidity and mortality, especially in susceptible groups such as young children, asthmatics, and those with chronic bronchitis and related conditions [9]. According to Streeton, J.A. [10], Nitrogen Dioxide (NO_2) appears to exert its effects directly on the lung, leading to an inflammatory reaction on the surfaces of the lung.

1.2.3 Volatile Organic Compounds (Hydrocarbons)

Volatile organic compounds contains a number of Hydrocarbons such as Benzene, Toluene, Xylene, 1, 3-Butadiene, Polycyclic Aromatic Hydrocarbons, Formaldehyde and Acetaldehyde. Some among them have fatal impacts on human body, carcinogenic effect in particular. Benzene and Polycyclic Aromatic Hydrocarbons are definitely carcinogenic. Some others such as 1,3-Butadiene, Acetaldehyde, Formaldehyde is likely to cause the same repercussion.

Xylene exposure has been associated with effects in a number of organ systems including the lungs, skin and eyes; neurological system; heart and gastrointestinal system; kidney; and possibly the reproductive system [11]. Inhaling toluene causes decreases in neurological function and irritation of the respiratory tract [12].

1.2.4. Sulphur Dioxide (SO_2)

Sulphur Dioxide (SO_2) is a strong respiratory irritant, and it has been associated with increased hospital admissions for respiratory and cardiovascular disease [13], as well as mortality [14]. Asthmatics form a particularly susceptible group when exposing to SO_2.

1.2.5. Ozone (O_3)

Ozone (O_3) is a powerful oxidant that can damage the respiratory tract, causing inflammation and irritation, and induces symptoms such as coughing, chest tightness, shortness of breath, and worsening of asthma symptoms. Exposure to Ozone (O_3) at the level above the current ambient air quality standard leads to lung inflammation and lung tissue damage, and a reduction in the amount of air inhaled into the lungs [15]. Recent evidence from research by

McConnell [16] has, for the first time, linked the onset of asthma to exposure to elevated ozone levels in exercising children.

1.2.6. Particulate matters

Studies have shown connection of exposure to particle pollution to a number of health problems including respiratory illnesses (such as asthma and bronchitis) and cardiovascular disease Moreover, chemical components of some particles formed as by products from the vehicle combustion process have been shown to cause cancer. These effects are often more associated with vulnerable groups such as the very young and the elderly. [17-20]

1.3. FUTURE RENEWABLE FUEL RESOURCES

In the mean time, Petrol is still the major vehicle fuel in transportation all over the world (80%). However, its downside impacts on human being and the environment, which have been known for decades, raise the need and urge scientists to explore new energy sources for future generations. A number of fuels, both fossil and non-fossil have been considered. Fuel types currently being recognised as alternative transport fuels are Electricity, Natural Gas (CNG), Propane (Liquefied Petroleum Gas – LPG), Ethanol, Methanol, BioDiesel and Hydrogen (H_2) [21]. Technically speaking, some among these fuels can be used in conventional engines with no or little modifications. Some others require completely different type of engines or control systems to the traditional ones to achieve the most desirable performance.

In the world of vehicles with alternative fuel choices they are generally classified into 2 categories:

- Dedicated vehicles

  Vehicles of this type are operated solely on an alternative fuel.

- Dual-fuel vehicles

  Vehicles belonging to this category are capable of being operated on both conventional and alternative fuel. This group is divided into 2 sub-categories:

♦ *Bi-fuel vehicles*

Vehicles can only be operated on one fuel at a time.

♦ *Flexible-fuel vehicles*

Vehicles can be operated on any mixture of the two fuels.

Among fuel types being used and investigated Hydrogen has gained great interest due to its abundance and renewable characteristics. Although Hydrogen does not appear in pure form in large quantity, it exists in water and is separated from Oxygen (O_2) through the electrolysis process. Hydrogen when used as fuel in Internal Combustion engines or Fuel Cells would produce much cleaner emissions than other fuels.

One of the disadvantages of Hydrogen fueled engines is the reduction in power by up to 30% with respect to gasoline-fueled engines. This is because Hydrogen fuel displaces the air in the combustion chamber.

The major barriers to mainstream Hydrogen usage are the implementation of fueling infrastructure and vehicle on-board storage. At standard temperatures and pressures, Hydrogen gas has a very low density. Therefore, it has to be compressed, stored in liquid form or in some other manner (e.g. bonded with other chemicals or stored in container with high pressure tolerance). Currently, each of these options has its drawbacks that can include safety, cost and the ability to store enough fuel onboard to provide acceptable traveling range. For example, to get reasonably high density when stored as a liquid, Hydrogen needs to be kept at -253°C at 2bar. The drawback is that it needs super-insulation to keep the boil-off losses (amount fuel lost to cooling) within acceptable limits, typically below 2% per day for vehicles.

The specific objectives of this thesis are to convert a 600cc 4-cylinder Kawasaki engine to a flexible-fuel engine capable of running on Hydrogen and Petrol. And then to develop an add-on electronic Hydrogen fuel injection system equipped with a Neural Network virtual sensor that can predict the vehicle emission components in real time.

CHAPTER 2

LITERATURE SURVEY

2.1. INTERNAL COMBUSTION (IC) ENGINE

2.1.1. Introduction to Internal Combustion (IC) engine

Internal Combustion engines have been designed to convert chemical energy in certain substances into useful mechanical energy by burning the substance in the presence of air. The ignitable substance is referred to as a fuel. An incredible amount of chemical energy is released in an explosion when the fuel-air mixture is ignited in the combustion chamber. The hot gas produced in this reaction rapidly expands and it is then transformed into mechanical work through the engine's mechanical mechanism. At the end of the conversion process the gas is discharged to the atmosphere.

An Internal Combustion (IC) engine is basically comprised of a combustion chamber (cylinder) and a piston. The piston reciprocates inside the cylinder. The linear motion is then converted to rotary motion by the crankshaft. The crankshaft is shaped to balance the pistons which are fired in a particular order to reduce engine vibration. The flywheel then helps smooth out the linear movement of the pistons. IC engines usually have more than one combustion chamber in their design and those types of engines are called multi-cylinder engine.

Internal Combustion (IC) engines are generally classified based on their cycle of operation or by their method of ignition. The cycle of operation refers to the number of strokes required for the completion of one engine power cycle. The method of ignition includes Spark Ignition (SI) and Compression Ignition (CI).

There are two cycles of operation: the two-stroke cycle and the four-stroke cycle. Two-stroke engines do not have valves as the four-stroke ones and fire once every engine revolution. Four-stroke engines fire once every two engine revolutions, hence generating less power than the 2-stroke types (2-stroke engines have twice as many power stroke per unit time than four-stroke engines). However, 2-stroke engines are rarely seen in cars due to the fact that rich Air/Fuel mixture is required because of exhaust gas residuals, resulting lower efficiency when comparing with another type. [22]

There are two modes of combustion: Spark Ignition (SI) and Compression Ignition (CI). SI engines use a device called spark plug to produce sparks to ignite the fuel mixture in the engine chamber. In CI engines, fuel is ignited spontaneously at high temperature and pressure during compression. CI and SI engines use different types of fuels. Diesel fuel is used in CI engines and has the characteristics of spontaneously combusting in the high pressure and temperature environments associated with piston compression. The conventional fuel used in SI engines is gasoline. Gasoline is manufactured to resist combustion under conditions of increased pressure and temperature (the condition of the engine during compression) since SI engines is designed to ignite its fuel at the right time only in the presence of an electronically discharged spark.

Four-stroke four-cylinder Internal Combustion (IC) Spark Ignition (SI) engines are the interest in this research and will be discussed in detail in the following sections.

2.1.2. Operation of a four-stroke SI IC Engine

The power of an internal combustion engine comes from burning a mixture of fuel and air in a small, enclosed space. When this mixture burns, it expands greatly, pushing the piston, thereby rotating the crankshaft. This motion is then sent to the wheels that move the vehicle [24].

An SI IC engine has one or more combustion chambers inside which a single piston moves with reciprocating motion (upwards and downwards). Each cylinder is fitted with a spark plug and at least two valves, the intake valve and the exhaust valve. A simple four-stroke SI IC engine is depicted in Figure 2.1. Each movement of the piston up or down the cylinder is one stroke of the four-stroke Otto cycle.

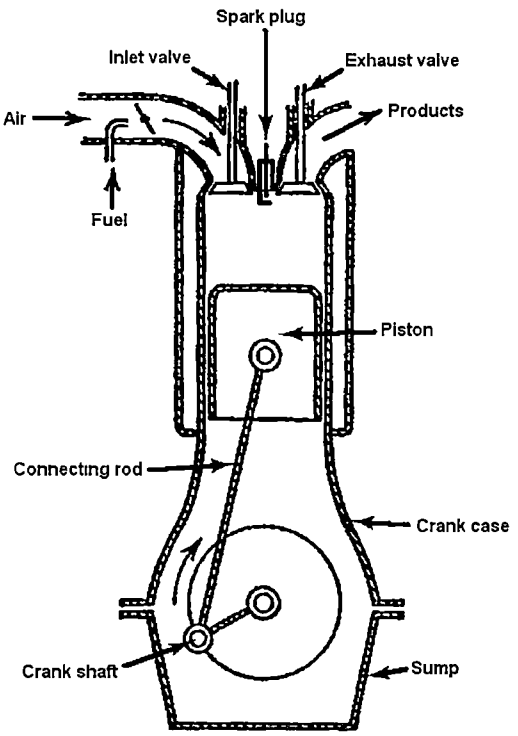


Figure 2.1: *Four-stroke SI engine* [23]

The four-stroke cycle of a SI IC engine is named after its inventor, Nikolaus August Otto. The Otto cycle (Figure 2.2) consists of induction stroke, compression stroke, ignition stroke (or power stroke) and exhaust stroke. [23]

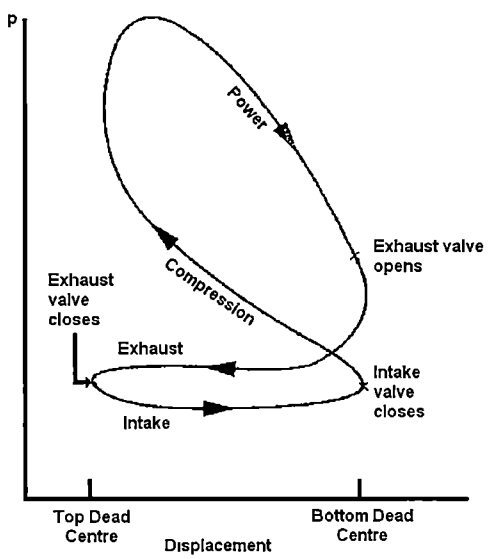


Figure 2.2: *Otto cycle of SI IC Engine* [23]

2.1.2.1. Induction stroke

During the induction stroke, the piston is moving downward and the intake valve is open. As a consequence of the movement of the piston a partial vacuum in the cylinder is developed. The air-fuel mixture is therefore forced into the cylinder past the open intake valve.

2.1.2.2. Compression stroke

At the end of the intake stroke the piston is at the lowest point of its movement, which is called Bottom Dead Center (BDC), and the intake valve closes. At this instance the engine combustion chamber is sealed. The piston then continues its travel going upwards and therefore compresses the combustible mixture inside the cylinder. Compressing the fuel mixture makes it even more combustible since not only the pressure in the cylinder but also the temperature of the mixture increases.

2.1.2.3. Ignition stroke

Ignition stroke is also referred to as the power stroke since it is the only stroke amongst the four that generates work to drive the engine crank shaft. As the piston reaches the top of the cylinder, also called Top Dead Center (TDC), the compressed air-fuel mixture is ignited by electrically discharged sparks from the spark plug. In burning, the mixture gets very hot and tries to expand in all directions. Since the piston is the only assembly that can move, the expanded gas forces the piston down. This force is carried through the connecting rod to the crankshaft and then through gears and shafts, turns the wheels of a vehicle.

2.1.2.4. Exhaust stroke

After the fuel is burnt, the exhaust valve opens while the piston is moving up to the engine Top Dead Center (TDC) and therefore the waste gases are forced out of the cylinder. The engine then repeats the first stroke (Injection stroke).

Engines in real world are comprised of more sophisticated elements varying with different designs. Fuelling system is one of those essential parts in an engine overall. In the next section, different fuelling systems for SI IC engines will be covered in detail.

2.1.3. Fuel injection systems

There are four main types of fuel injection systems in automotive application running on gasoline: carburetor, single-point (throttle body) injection, multi-point (port) injection and direct injection. These systems all employ different methods of providing the combustion chambers of the engine with an air and fuel mixture in a form in which it can be burnt.

2.1.3.1. Carburetor

Among all the types of fuel injection systems, carburetor has the longest history of being used in engine design. It is essentially a device that mixes air and fuel together in correct amounts and proportions for all different operating conditions of the engine and makes use of engine vacuum to feed fuel into the engine.

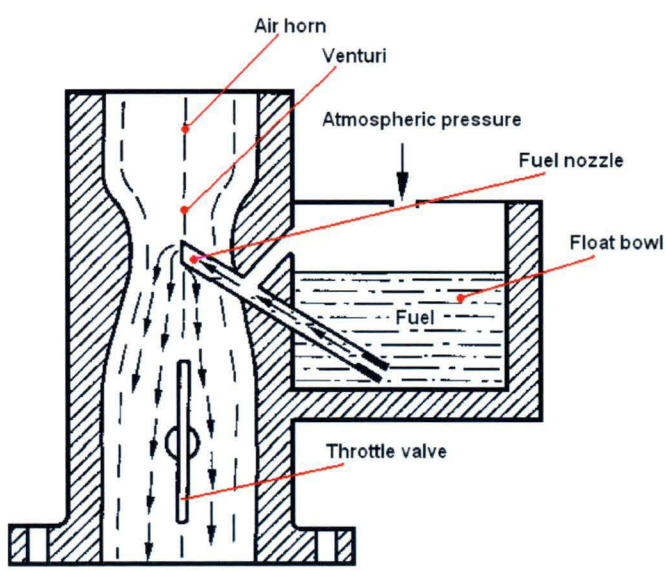


Figure 2.3: *Carburetor working principle* [25]

In an IC engine, the carburetor is attached to the engine intake manifold and an air cleaner is fitted onto the top of the carburetor to trap dirt that may cause damage to the cylinder wall and the piston.

The basic working principle of a carburetor is described in Figure 2.3. A simple carburetor consists of the following components: [25]

- Float bowl
  The float bowl holds the fuel in the carburetor

- Fuel nozzle
  The nozzle discharges the fuel.

- Air horn
  The air horn is also referred to as the throat or barrel. It directs ambient air into the engine intake manifold.

- Venturi
  The venturi produces sufficient suction to pull fuel out of the fuel nozzle. Air flows through the venturi by the suction effect of the piston during the induction stroke.

- Throttle valve
  The throttle valve is connected to the accelerator pedal of the vehicle. It controls the amount of combustible mixture delivered into the engine chamber and thus controls power and speed of the engine as well.

The atmospheric pressure applies onto the surface of the fuel in the float bowl forces the fuel from the nozzle to disperse into the low-pressure area, which is at the constriction in the venturi. The low-pressure area in the venturi is caused by the air flowing through it. Due to the fact that air consists of molecules, a high-pressure area is formed when those molecules are crowded together and a low-pressure region will appear when those molecules are separated.

After entering the venturi, air molecules must leave it with the same rate. Therefore, there is apparently a temporary increase in speed of those molecules and thus results in the effect of separating the air molecules. The separation then forms a low-pressure area in the venture where the air molecules reach their highest speed. The low-pressure area forces the atomised fuel from the fuel nozzle to discharge [25]. The fuel molecules are therefore mixed with the air stream to form the combustible mixture and then supplied to the engine combustion chamber. The higher the flow rate of the air flowing, the lower will be the pressure at the fuel nozzle and thus the greater will be the amount of fuel poured out at the nozzle.

The throttle valve controls the airflow in the carburetor. Airflow gets to maximum flow rate when the throttle valve is fully opened. At that instance the carburetor delivers maximum amount of fuel mixture into the cylinders.

Nowadays, carburetor injection system is replaced with electronic fuel injection system. The major drawback of carburetor is that the amount of fuel drawn from the nozzle is not proportional to the quantity of airflow inducted into the engine at different speeds. A number of modifications have been added to the carburetor's basic design to improve the accuracy in controlling air/fuel ratio but then makes it an extremely complicated mechanical mechanism.

There are many types and configuration of injection systems that employ electronic timing method. Those injection systems have several advantages over a carburetor type of fuel system such as: [26]

- Improved atomisation
  Fuel pump provides fuel into the intake manifold under pressure and that helps break fuel droplets better.

- Better fuel distribution
  Fuel injectors have a common fuel rail and therefore equal flow of fuel vapors is expected to be sprayed into each cylinder.

- Smoother idle
  Lean fuel mixture can be used without rough idle due to better fuel distribution and good low-speed atomisation of fuel droplets.

- Lower emissions.
  Fuel capable of being mixed with air at lean and stoichiometric air-fuel ratio reduces exhaust pollution (stoichiometric ratio is the theoretically ideal ratio in which fuel would be completely burnt and no oxygen will be left after the combustion process).

- Better cold weather drivability.
  Electronic timing provides better control of mixture enrichment than a carburetor.

- Increased engine power.
  Precise control of the amount of fuel delivered to each cylinder and increased airflow can result in more horsepower output.

- Fewer parts.
  Electronic timing injection systems apparently consist of fewer and less complicated parts than carburetor systems.

The following sections would cover 3 main types of modern fuel injection system available in the market: indirect injection and direct injection systems.

2.1.3.2. Indirect injection systems

There are 2 types of indirect fuel injection systems: Throttle body injection and Port injection. In those set-ups fuel is sprayed into the intake manifold. Unlike a carburetor, instead of making use of engine vacuum, pressure is used to feed fuel into the engine. This therefore makes those injection systems more efficient than a carburetor.

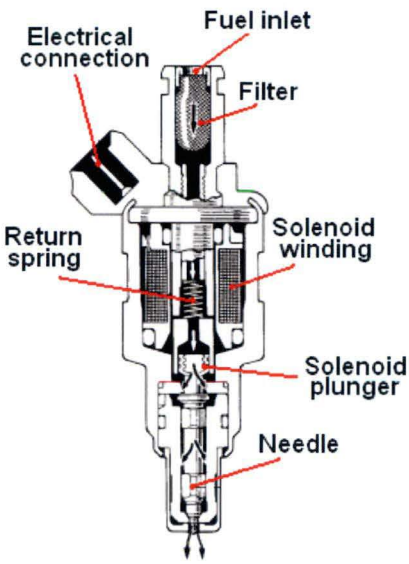


Figure 2.4: *Basic electronic fuel injector* [25]

In injection methods that do not use carburetor, fuel injector is an essential part in the system. Fuel injector is used to spray fuel into the mixing port and the electronic designs work on the principle of a solenoid valve (see Figure 2.4). When an electrical current passes through the solenoid winding the needle is lifted and hence releases the fuel at the same pressure of the inlet stream. If the electrical current is cut, the return spring forces the needle to go back to its original position and closes the outlet.

2.1.3.2.1 Throttle body injection system

A throttle body injection (TBI) system, which is also referred to as single-point injection system, has the injector nozzles in a throttle body on top of the engine as shown in Figure 2.5. Fuel is sprayed into the top center of the intake manifold [27]. This type of injection system may have one or two fuel injectors in the throttle body assembly [25]. Fuel is supplied to the fuel injectors under pressure.

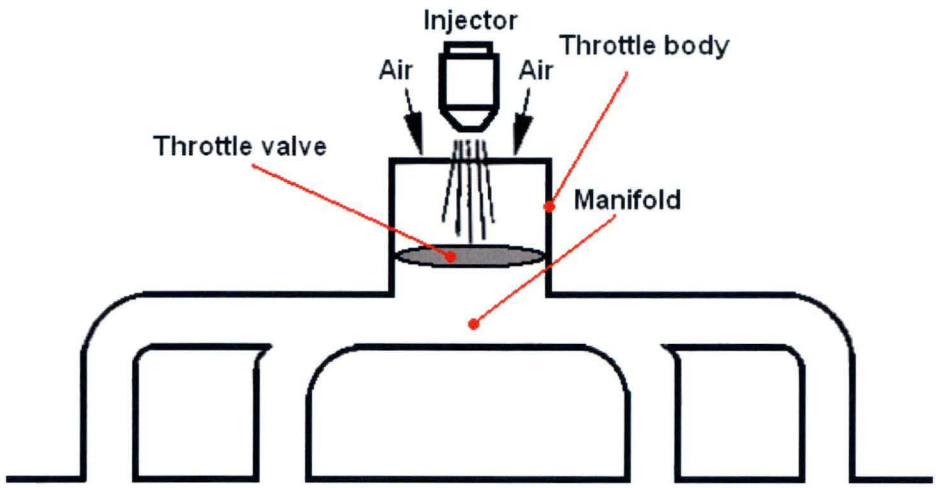


Figure 2.5: *Throttle body injection system (Single point injection)* [25]

When the injector opens, fuel is sprayed into the throttle body and mixed with the air flowing through the assembly. The mixture then passes through the intake manifold to the engine chamber. The amount of airflow is controlled by the throttle valve. In turn, it controls engine power output. Like the carburetor throttle valve, it is connected to the engine gas pedal. When the pedal is pressed, the throttle valve swings open to allow more air to be inducted into the engine. Timing of the injector is controlled by the engine control unit (ECU) to ensure correct air/fuel mixture is delivered to the engine over the range of operating condition.

2.1.3.2.2 Port injection

Port injection system shown in Figure 2.6 is also called multipoint injection due to the fact that fuel is sprayed into each intake port and towards each intake valve (more than one location). An electric fuel pump draws fuel out of the tank and forces it into the pressure regulator. The pressure regulator maintains the pressure in the fuel lines and controls the amount of pressure entering the injector valves. When sufficient pressure is reached, the regulator returns excess fuel to the tank.

The throttle valve regulates how much air flows into the engine and hence controls the power the engine delivers. Control timing to synchronise operation of all the injectors is done by the

central engine control unit (ECU). When the injector opens, fuel is squirted into the intake manifold under pressure

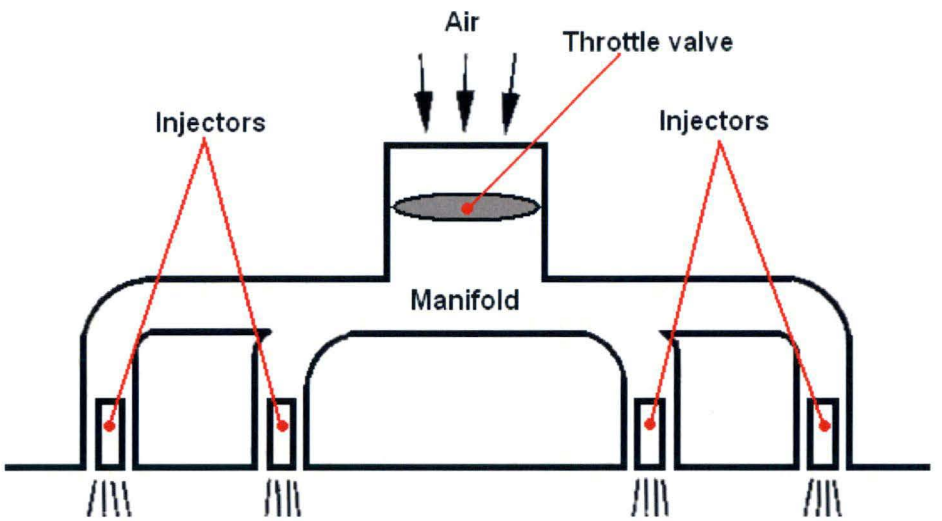


Figure 2.6: *Port injection system (multipoint injection system)* [25]

The location of a fuel injector in the port injection set-up is shown in Figure 2.7. It is fitted into the intake manifold so that the fuel is sprayed directly into the intake port towards the cylinder. The angle of fuel spray is about 25^0 [25].

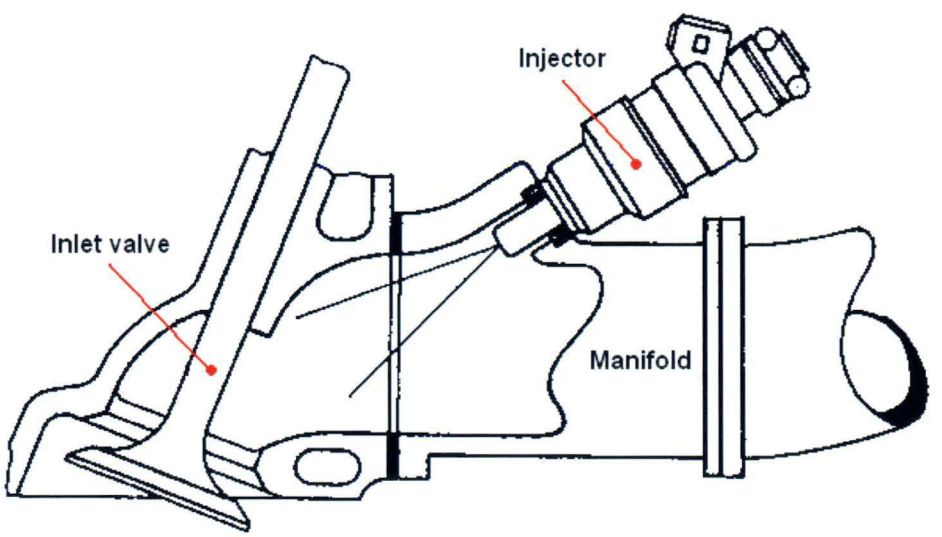


Figure 2.7: *Fuel injector location in port injection system* [25]

2.1.3.2. Direct injection system

While carburetor and indirect fuel injection systems are conventionally employed in Gasoline engine designs, direct injection method can be seen in both Diesel and Gasoline engines.

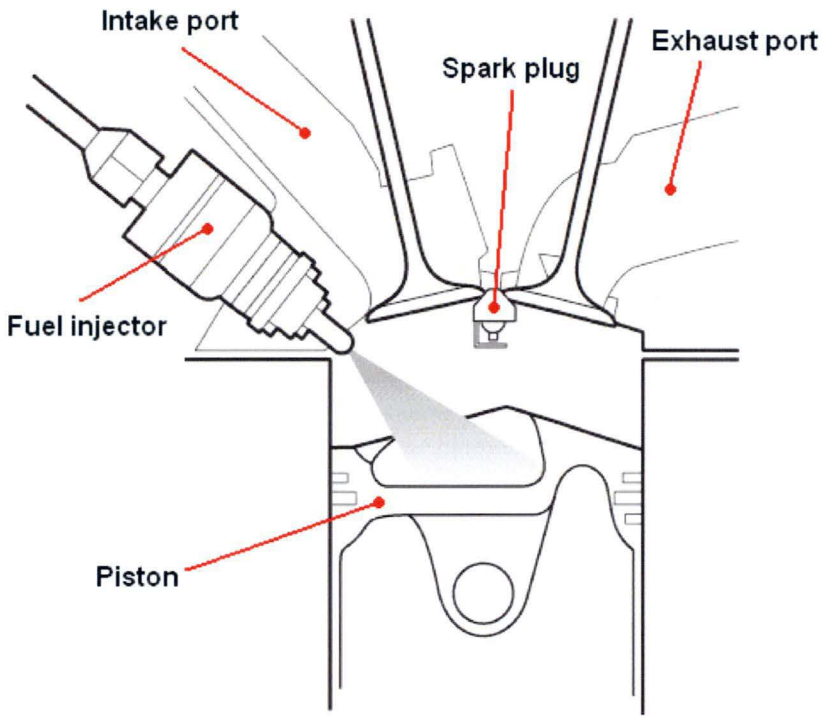


Figure 2.8: *Gasoline direct injection (GDI) engine* [28]

Gasoline direct injection (GDI) technology engine has drawn enormous attention over the last few years as a major step in development of SI engine design. As shown in Figure 2.8, the design has an injector mounted into the cylinder head combustion chamber instead of being mounted elsewhere on the intake port. In this type of injection, the fuel supply system has to deliver fuel at higher pressure than the pressure in the port injection configuration. It is because the fuel injector nozzle is embedded inside the combustion chamber and depressed by the in-cylinder compressed air.

There are two modes of operation in GDI engine: homogeneous and stratified charge. [29]

- Homogeneous charge

  Fuel is injected during the intake stroke. The fuel mixture ratio in this mode is near the stoichiometric value. Therefore the engine conveys significant torque to the output shaft.

- Stratified charge

  Fuel is injected during the compression stroke. This mode is suitable for cruising and light load conditions.

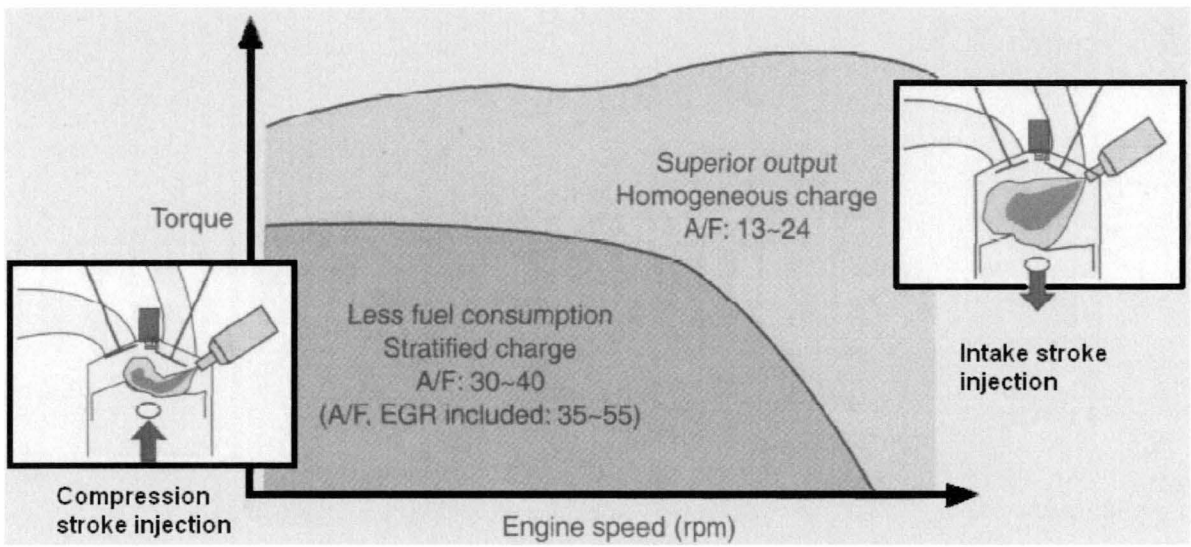


Figure 2.9: *Combustion modes of GDI engine* [29]

Running the GDI in stratified charge mode results in higher thermal efficiency (lower fuel consumption and lower CO_2 emissions) comparing with an engine with port fuel injection as shown in Figure 2.9. However the overall fuel lean mixtures can increase the formation of NO_X [28].

2.2. DUAL FUEL INJECTION SYSTEM

Governments around the world are currently enforcing stricter rules on emission requirements especially from transportation means. In response to tougher standards in global exhaust gas control, various dual fuel systems for internal combustion engines have been introduced by the automotive industry that couple Petrol or diesel with alternative fuels such as a compressed natural gas (CNG) – Petrol system, a compressed natural gas (CNG) – diesel system or a liquefied Petroleum gas (LPG) – Petrol system... Those dual fuel technologies have claimed to dramatically reduce the harmful emissions and are becoming friendlier to the environment.

There are three main streams in research and development of dual fuel systems. The first one is modifying the carburetor to allow the engine to induct a homogeneous mixture of either air and Petrol or air and the alternative fuel. The second trend that is more popular is manufacturing and modifying the injection port to convert conventional engines to run on different composition of Petrol and the added fuel. This method is also suitable for designs of SI IC engines running on dual combination of fuels other than Petrol such as CNG and alcohol. The final approach is using fuel injectors that are capable of delivering more than one type of fuel into the engine combustion chamber. This technique is applied only to direct injection engines such as diesel engines or Petrol engines originally designed with direct injection.

In this research, four types of dual fuel injection systems are investigated in detail. They are compressed natural gas (CNG) – Petrol engine with carburetor, liquefied Petroleum gas (LPG) – Petrol engine with carburetor, modified port dual fuel injection system to facilitate Hydrogen-Petrol fuel and diesel-natural gas system equipped with dual fuel injectors.

2.2.1. Compressed natural gas (CNG) – Petrol dual fuel system with carburettor

In a compressed natural gas (CNG)-Petrol dual fuel injection system the gas is compressed to a very high pressure but, unlike LPG, is not liquefied (see Figure 2.10). The Petrol fuelling system is the standard one. The Petrol/gas selector enables the driver to choose the fuel for engine operation. When CNG is required, it first passes through the first-stage regulator to reduce the high pressure to a usable level that is within the tolerance of the second stage of regulation. After being regulated by the device named second- and third-stage regulator the CNG gas has a serviceable pressure ready to deliver to the engine.

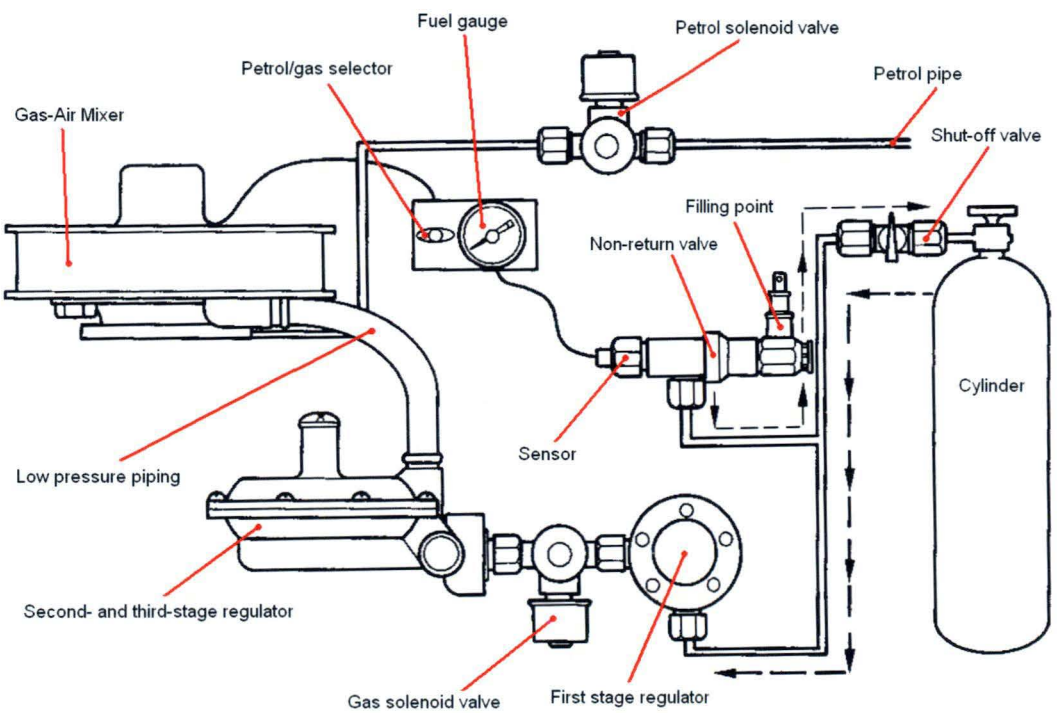


Figure 2.10: *Basic compressed natural gas (CNG)-Petrol dual fuel injection system* [25]

The gas-air mixer is mounted on top of the original Petrol carburetor. It meters and mixes the gas and air before the mixture is inducted into the engine via the carburetor. The throttle valve in the carburetor determines the amount of gas flowing into the engine chamber as when Petrol is used.

2.2.2. Liquefied Petroleum gas (LPG) – Petrol dual fuel system with carburetor

A simplified diagram of an LPG fuel system for a motorbike engine with carburetor is shown in Figure 2.11. When the engine is in idling state the fuel is supplied straight to the venturi of the carburetor. When the valve float moves up as the throttle grip is twisted, the system delivers LPG fuel to the carburetor. As it can be seen in the layout the amount of fuel being supplied increases with the degree of twisting at the engine throttle grip. In other words, more fuel is delivered as more power is required.

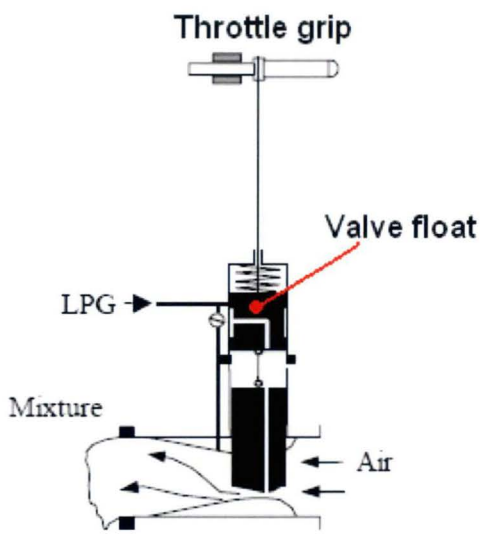


Figure 2.11: *Basic motorbike LPG fuelling system with carburetor.* [30]

Using the basic LPG fuelling schema shown above, researchers from Vietnam have achieved significant success in converting conventional motorbikes to run on dual fuel, Petrol and LPG. The principle of conversion is modifying the original motorbike carburetor to facilitate both fuels to keep the cost of the conversion process low and affordable to the major population.

When using LPG as a fuel, it is stored in liquid form in high pressure cylinders. Before it is supplied to the fuelling system, the fuel pressure has to be reduced to transform it to gaseous state. Pressure reduction process is achieved by using a device called a converter or a vaporiser-regulator and it also ensures that the fuel is vaporised before being fed into the

engine [25]. The LPG gas is aspirated to the venturi by depression in the same way of inducting gasoline by carburetor.

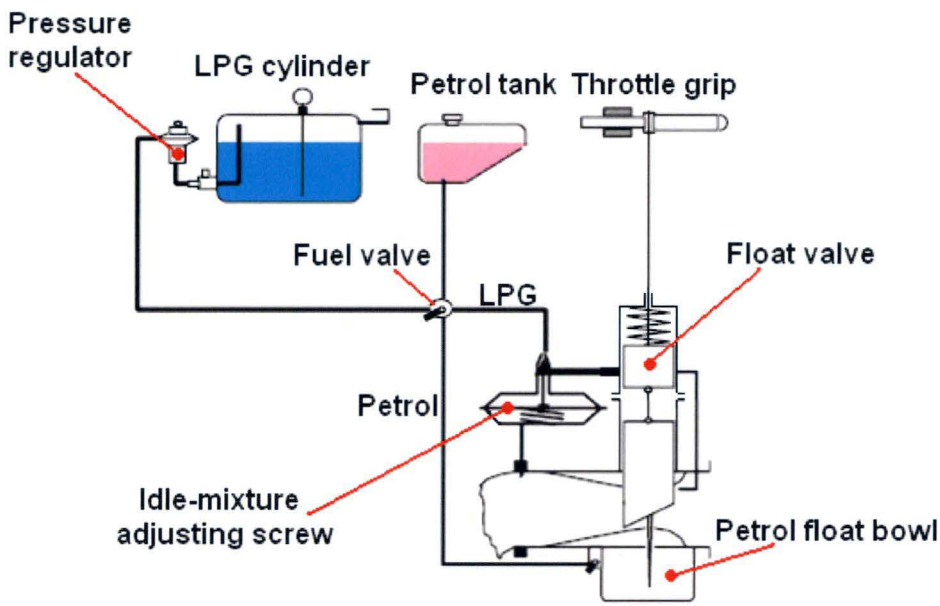


Figure 2.12: *Motorbike liquefied Petroleum gas (LPG)-Petrol dual fuel injection system.* [30]

Figure 2.12 depicts an LPG-Petrol dual fuel injection system converted from the conventional carburetor of a motorbike. Due to limited space on motorbikes it is preferable to avoid using the vaporiser device to reduce fuel pressure [30]. Thus the most suitable approach is supplying LPG in gas state with pressure slightly higher than that of the atmosphere [31] [32].

The fuel valve acts as a switch to divert the right fuel of choice into the modified carburetor. When the valve is at the Petrol position, Petrol is delivered into the Petrol float bowl and then aspirated into the venturi exactly in the same way as the original carburetor. When the switch is turned to the LPG position it allows LPG to be supplied to the fuelling system. In the idle mode LPG gas passes the idling system with the adjustment screw. More LPG fuel is delivered into the system when the throttle grip is twisted thus resulting more power output from the motorbike engine.

2.2.3. Modified port dual fuel injection system

Since the day multi-point injection system being introduced by the automotive manufacturers, together with rapid development of the electronic engine control industry, electronically timed fuel injection has become a popular method in engine design. In such approach fuel is supplied to fuel injectors via a common fuel line, which is also referred to as fuel rail. The number of fuel injectors depends on the number of cylinders of the engine and they are all controlled by a central computer (Figure 2.13). The computer synchronises timing and determines when each injector opens to spray fuel into the intake manifold or closes to stop forming the fuel mist.

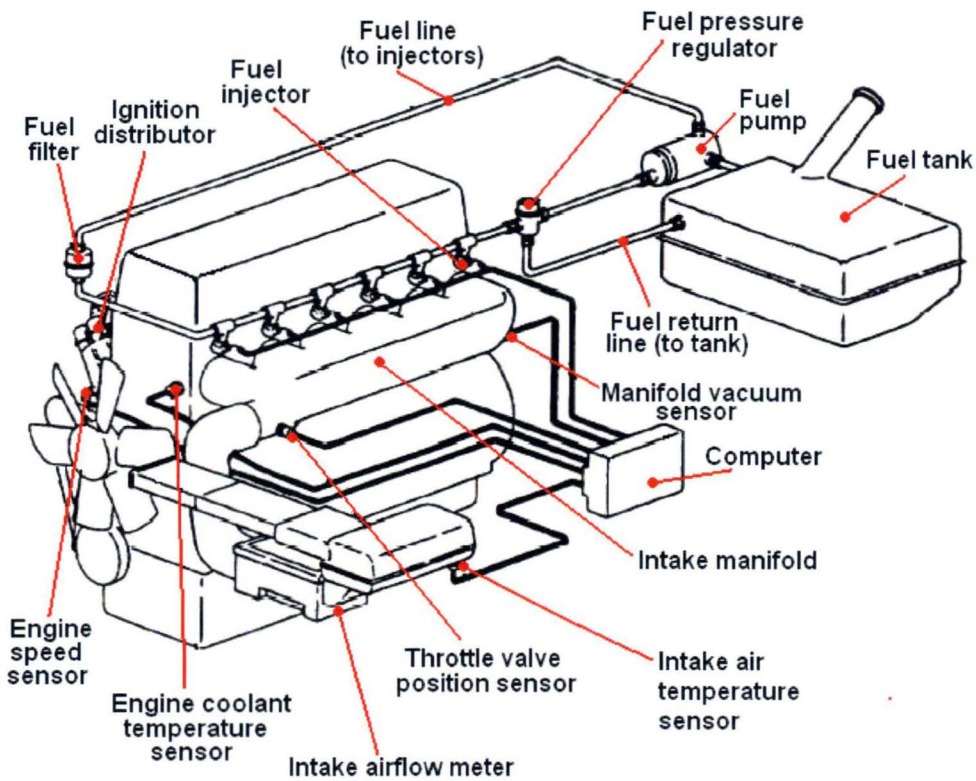


Figure 2.13: *Electronic timed fuel injection system.* [26]

With such configuration of injection, when converted to a dual fuel system the intake manifold needs to be modified to facilitate mounting of injectors and fuel rail of the alternative fuel. The storage gear, pressure regulation device and supply piping for the additional fuel depend on the

type of the fuel (CNG, LPG, Hydrogen…) and have to meet safety standards for that specific energy source.

2.2.4. Diesel-natural gas system with dual fuel injectors

There have been significant efforts by the industry and researchers around the world to reduce the emissions by conventional diesel engines. Since Diesel engine is basically a direct injection engine, the preferred conversion method is to replace the original injectors with injectors that are capable of delivering more than one type of fuel into the engine chamber. High Pressure Direct Injection from WESTPORT Inc [33] is a typical example.

"The system relies on late-cycle high-pressure injection of a gaseous fuel, such as natural gas, into a combustion chamber of an internal combustion engine. The natural gas is injected at the end of the compression stroke, just like the diesel fuel is injected at the end of the compression stroke in a diesel engine. Under the pressures found in the combustion chamber of a normal diesel engine, natural gas requires higher ignition temperature than diesel (circa 800° C vs. 500° C) to maintain acceptable ignition delay period of less than 1 millisecond. To assist with the ignition of natural gas, a small amount of diesel fuel is injected into the engine cylinder using the same injector followed by the main natural gas fuel injection as shown in the Figure 2.14. This diesel fuel acts as a pilot or "liquid spark plug" which ignites rapidly the hot combustion products then igniting the natural gas". [33]

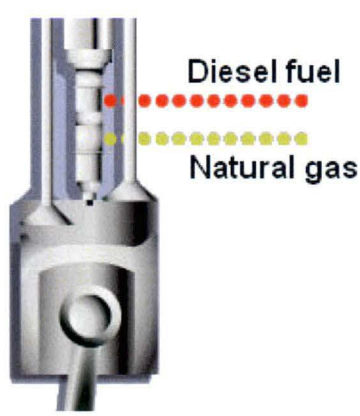


Figure 2.14: *High Pressure Direct Injection system.* [33]

Figure 2.15: *High Pressure Injection dual-concentric needle injector.* [34]

The dual-concentric needle injector (Figure 2.15) is controlled electronically and enables small quantities of diesel fuel and large quantities of natural gas to be supplied at high pressure to the combustion chamber. The diesel fuel is delivered just before the piston reaches its top dead centre, followed by the main fuel quantity of natural gas.

Such fuel injectors are designed to manage: [34]

- The high volumetric flow rates required for a fuel much less dense than diesel
- Complex fuel routing for two fuels within the injector
- Carbonising and thermal fluctuations that can result from the reduced cooling capacity of a gaseous fuel
- High pressure differentials within the injector and fuel leakage that can result

2.3. IC ENGINE CONTROL

Internal combustion engine itself is a mechanical system that converts chemical energy sources to useful power. However, in order for it to function and operate properly, associated control instrumentation is required.

Internal combustion (IC) engine control is among the most complex problems for the automotive industry. Due to the increasing requirements of legislations by governments around the world, car manufacturers always strive to substantially reduce emissions and fuel consumption while maintaining the best engine performance. To satisfy those requirements various variables need to be controlled, such as engine speed, engine torque, spark ignition timing, fuel injection timing, air intake, and air-fuel ratio… [35] Each of those variables is related to others in a complicated manner. In addition to that, vehicle engines have a number of different operating modes including start-up, idle, running, and braking. Therefore engine control dynamics are highly non-linear and multivariable.

Engine fuel injection and spark ignition are controlled by a micro-controller device called Engine Management System (EMS) or Engine Control Unit (ECU).

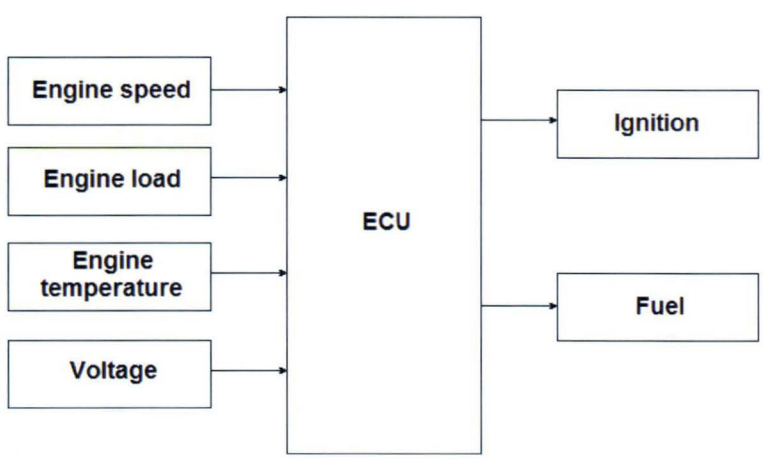


Figure 2.16: *General block diagram of an automotive Engine Management System* [36]

Figure 2.16 above illustrates a general block diagram of an Engine Management System with basic sensor inputs and control outputs.

Engine management in general is the technology of employing current achievement in electronic control science to calibrate an engine to achieve the cleanest possible exhaust stream and at the same time maintaining fuel economy and the most desirable performance. Furthermore, it also includes continuous diagnosing of system faults within the control boundary. The focus on the aspects mentioned above varies worldwide depending on government regulations, customer expectations and driving conditions...[37]

Modern Engine Management System also delivers control methodology to other engine subsystems such as lambda control, evaporative emission control, knock control, Exhaust Gas Recirculation (EGR) control, Anti-lock Brake System (ABS) control, traction control and stability control...

The scope of the following discussion will be limited to engine fuel injection and ignition timing in an engine control system.

2.3.1. Conventional electronic engine control methodology

In a conventional electronic Engine Management System controlling an engine with multi-point injection set-up, various data from a number of sensors is processed to determine the correct amount of air and fuel that would be delivered to the engine cylinder and the ignition advance angle at different engine operating condition. Advance angle is a measure of how early the sparking takes place before the engine piston reaches its Top Dead Center (TDC) starting the power stroke.

Besides dedicated engine management products manufactured for vehicle manufacturer, there is a wide range of off-the-shelf engine control systems in the market designed for amateurs or the racing industry such as MoTeC, HalTek...

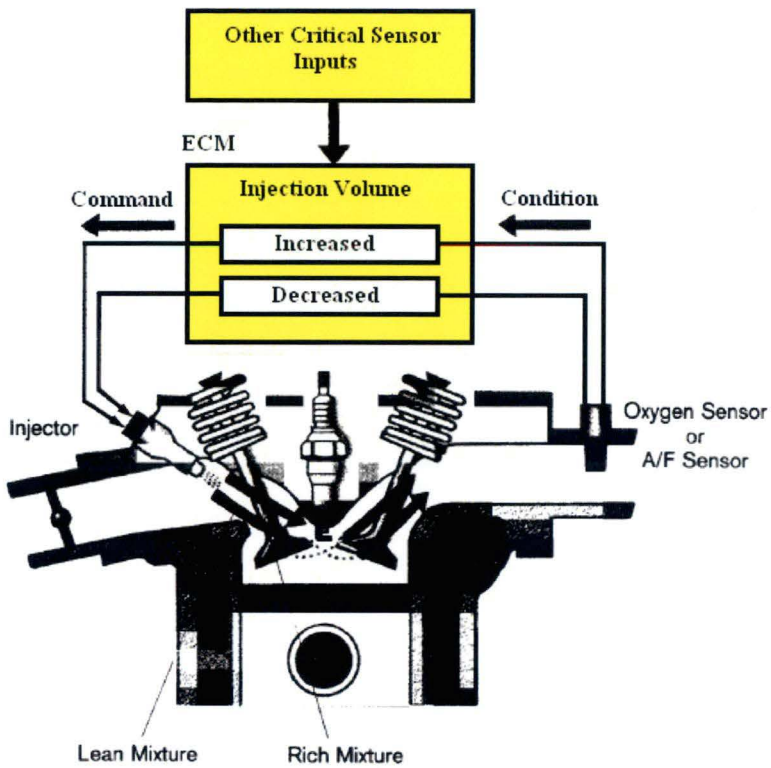


Figure 2.17: *Closed loop Engine Management System* [38]

There are two modes of operation in a conventional Engine Management System: the closed loop and the open loop as shown in Figure 2.17.

Control of ignition in a typical Engine Management System is usually implemented in an open-loop manner. It means the ignition control system does not monitor its output and make correction based on its output.

Injection control however may involve both operation modes. The Engine Control Unit controls the injection with the open-loop algorithm when: [38]

- Starting the engine
- Engine being cold
- Hard acceleration
- During fuel cut-off
- Throttle being widely opened

Legislations and regulations on emissions require the existence of closed-loop fuel control systems on modern vehicles. The Engine Management System monitors the exhaust stream and regulates control parameters to maintain the ideal lambda value, which represents the stoichiometric air/fuel ratio (14.7:1), to help the catalytic converter operate at peak efficiency.

Closed-loop operation controls the average air-fuel ratio of all cylinders [39][40]. There are two types of lambda sensors being used in feedback control applications: the switching Exhaust Gas Oxygen (EGO) sensor and the Universal Exhaust Gas Oxygen (UEGO) sensor. Both types of sensors analyse the air/fuel ratio based on the exhaust composition and operate at the temperature range of 250 – 300°C therefore closed loop operation is only initiated when this temperature is reached.

2.3.1.1. Closed-loop operation with Exhaust Gas Oxygen (EGO) sensor

When in the closed-loop mode, the Engine Management System uses the signal from the EGO sensor to make correction to the injection duration so that the catalytic converter can work at peak efficiency. Each EGO sensor has a threshold voltage. When the voltage it generates is higher than the threshold the air/fuel ratio is judged to be richer than the ideal air/fuel ratio. Otherwise, the air/fuel ratio is judged to be leaner than the ideal one.

When the mixture is richer than the stoichiometric air/fuel ratio, the engine control algorithm reduces the amount of fuel injected at a constant rate. The reduction continues until the EGO sensor switches to the low voltage. In contrast, when the voltage signal is lower than the threshold meaning the air/fuel ratio is judged to be leaner than the ideal number the amount of

fuel injected is increased at a constant rate until the EGO sensor voltage switches to the high level.

In other words, the closed-loop system with EGO sensor is operating in a switching manner and therefore the air/fuel ratio oscillates richer or leaner from the ideal ratio. [38]

2.3.1.2. Closed-loop operation with UEGO sensor

The UEGO sensor is slowly replacing the EGO sensor in modern automotive application due to its ability to give the actual air/fuel ratio [41]. With this type of sensor air/fuel mixture correction done by the vehicle Engine Management System is more precise and quicker since the voltage it generates is proportional to the mixture air/fuel ratio [38]. The control system knows how much the actual air/fuel ratio is deviated from the stoichiometric ratio then quickly adjusts the injection duration.

In conclusion, lambda sensor of EGO and UEGO types are used in closed-loop fuel control system during steady state and idle speed condition. When in this mode, the Engine Management System uses Proportional (P) or Proportional Integral (PI) control algorithm to maintain stoichiometric ratio of the air/fuel mixture. [39][40]

2.3.1.3. Tabular map data modeling

As mentioned above, the relationship between engine load, engine speed, engine fueling and ignition is non-linearly complex. The engine load is either represented by the throttle position or the manifold absolute pressure. Therefore data is presented in forms of 3D maps to co-ordinate all those variables to determine the appropriate amount of fuel and the right instance of ignition.

The three-dimensional maps are as shown in Figure 2.18 and 2.19. They are hard-coded into the memory residing in the Engine Management System and are also referred to as look-up

tables. The map in Figure 2.18 represents the fueling requirements and Figure 2.19 shows the advance angle for ignition control corresponding to the whole range of engine operation.

Either the required air/fuel ratio (λ), the ignition advance angle or the injection duration is on the z-axis of the map and each of them is mapped as a function of both engine speed and load which are located on the x and y-axis. The Engine Management System simultaneously calculates the speed and reads in the data from load sensors and then looks into the previously stored maps for the base injection opening pulse width and the ignition advance timing angle for the current engine operating condition. It then sends control signals to the related actuator control modules.

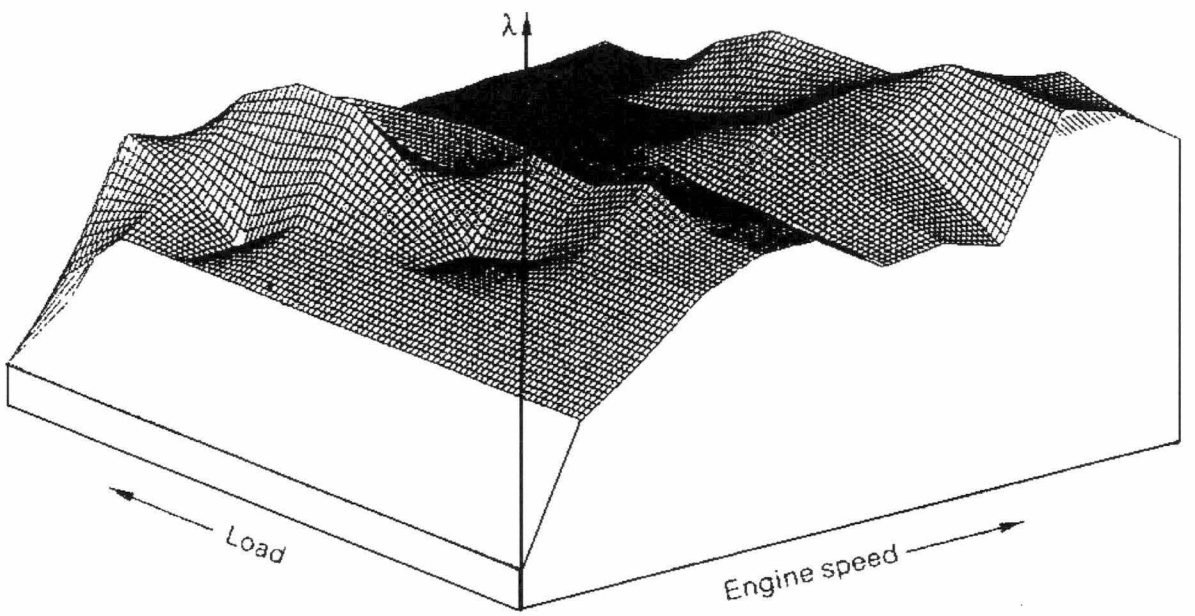


Figure 2.18: *Three-dimensional fuel requirement map* [36]

Linear interpolation is utilised to extract data from look-up tables when engine operating condition is falling in between points in the map. The accuracy is therefore depending on the resolution of the table grid and the degree of non-linearity. [42]

In order to establish the maps that represent the non-dynamic modeling and non-linear relationships of parameters within an engine numerous testing and tuning over the whole range of engine operation on engine dynamometer are required. In convention, the tuning processes

would help construct a map that stores the start of injection data, a map storing the duration of injection data and a map storing the advance ignition angle. Those maps store the base data and represent the optimum air/fuel mixture that satisfies the power demand and smooth operation requirement while at the same time meeting the gas emission standards. Corrections to those base values are required since engine control is a highly complex and non-linear process and involves many other external parameters. Firstly, the Engine Management System corrects for air temperature. As the ambient temperature changes the air density and hence the air mass that enters the cylinder changes. Secondly, data is corrected to compensate for after-start enrichment (choke), acceleration enrichment, weakening on deceleration, cut-off on overrun, reinstatement of injection after cut-off and correction for battery voltage variation...[36].

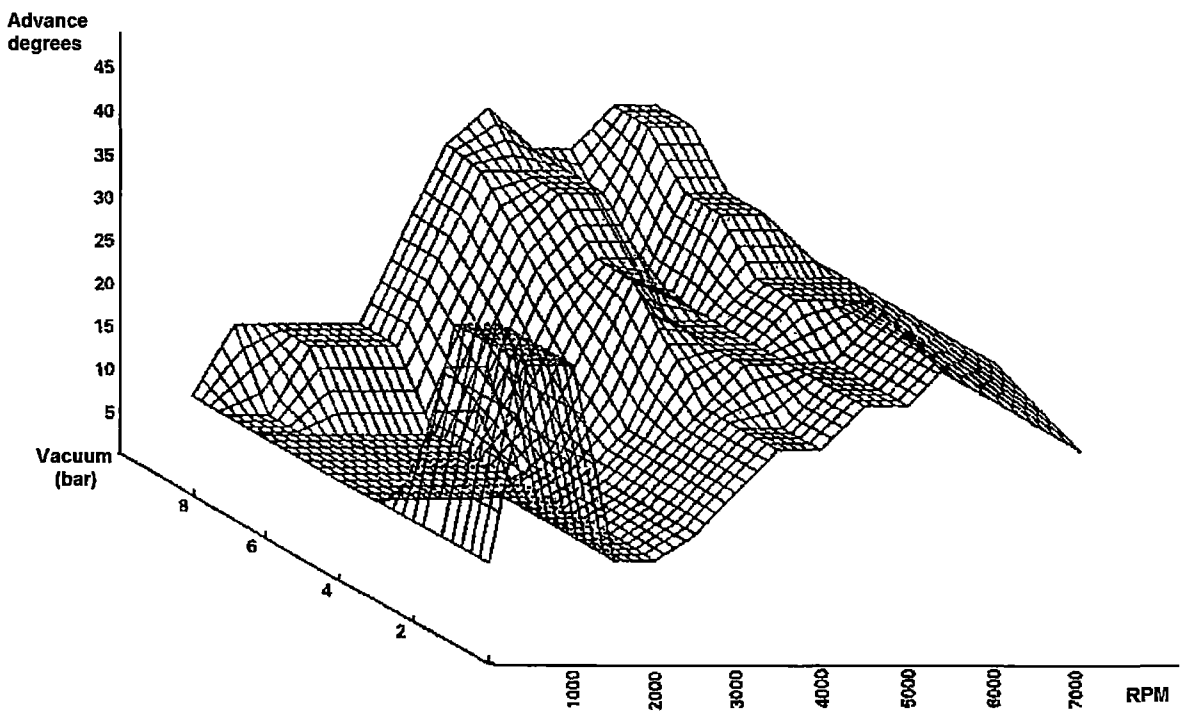


Figure 2.19: *Three-dimensional advance angle map.* [23]

The correction factors are also developed during tuning and stored in the Engine Management System in the form of three-dimensional look-up maps. In modern vehicle an engine control system may store about fifty maps to model the non-linearity aspect of automotive control.

2.3.2. Engine control using modern techniques

Current open-loop strategy such as using look-up tables is inflexible and inaccurate for highly non-linear IC engines compared with closed-loop control [43-46]. However closed-loop control methods cannot be used efficiently for engine variables due to the fact that creating an analytic engine model with high accuracy is a difficult process [47].

Since conventional engine control methods retain problems, researchers around the world have been studying and researching advanced control methodologies in recent years and artificial intelligence is one of the promising fields that have been investigated. The following sections summarises several representative examples of applying Artificial Intelligence in control of IC engines.

2.3.2.1. Diesel engine control with Local linear radial basis function network

In the work by Hafner et al [48], the application of a local linear radial basis function network in modeling for engine control is presented. The neuro-models are integrated into an upper-level emission optimization tool that calculates a cost function for exhaust versus consumption/torque. Based on that it determines the optimal engine control settings. According to this research the developed system allows fast application of the optimisation tool at the engine test stand.

2.3.2.2. Control for idle speed regulation in IC engines with neural network

Idle speed control is critically required in engine control. It stablises the engine speed at idle while engine parameters are changing in a heavily non-linear manner. Therefore an adaptive sliding mode control design method is proposed in the research by Li et al [49]. This method is usually used for discrete non-linear systems where explicit knowledge of the system dynamics does not exist. Three–layer feed-forward neural networks are used as function approximator for the unknown dynamics. The control law is designed based on the outputs of the approximators, and the sliding surface is defined in terms of a stable polynomial of the system outputs.

2.3.2.3. Spark advance control using cylinder pressure and neural network

Spark advance control using the location of peak pressure (LPP) as feedback combining with Aritifical Neural Networks is demonstrated in the work by Park et al [50]. The problems retained in the LPP-based spark advance control method are excessively huge amount of data samples is required and difficulty in detecting the combustion phasing owing to hook-back during lean burn operation. Therefore a feed-forward neural network is developed to help estimate the LPP and hook-back and only five samples of pressure data are required in a cycle. The feed-forward controller consists of a Radial Basis Function Network and a feed-back error learning method is used for the training. The feasibility of this method is proven through steady and transient engine operations.

2.4. EMISSIONS FORMATION

During engine operation, different species of gases emit and disperse into the atmosphere. Those gases include carbon monoxide, nitrogen oxides, hydrocarbons, sulphur oxides, ozone and others as by-products. This section briefly investigates the formation of those specific gases.

- *Carbon monoxide*

  Incomplete combustion of carbon containing fuels, including Petrol and Diesel is the main cause of formation of Carbon monoxide. Running engine on lean mixture would reduce the amount of this gas.

- *Nitrogen oxides*

  Nitrogen oxides (primarily nitric oxide and lesser quantities of nitrogen dioxide) are gases formed by oxidation of nitrogen in air at high combustion temperatures. Nitric oxide is oxidised to nitrogen dioxide in ambient air, which plays a major role in chemical reactions that are linked to the formation of photochemical oxidants such as ozone and particles such as nitrates.

- *Hydrocarbons*

  Hydrocarbons exist in engine exhaust gases mainly due to excessive fuel being injected into the engine. Volatile organic compounds are a range of hydrocarbons, the most important of which are benzene, toluene, and xylene, 1,3-butadiene, polycyclic aromatic hydrocarbons (PAHs), formaldehyde and acetaldehyde.

- *Sulphur oxides*

  Sulphur oxides (primarily sulphur dioxide and smaller amount of sulphur trioxide) are gases formed by the oxidation of sulphur contaminants in fuel during engine combustion.

- *Ozone*

  Ozone is formed by reactions of nitrogen oxides and volatile organic compounds in the presence of sunlight.

2.5. EMISSIONS PREDICTION IN IC ENGINE

IC engines, or vehicles in specific, are being required to comply with increasingly stringent government exhaust emissions regulations. In addition to that, engines in the future will require significantly more complex control than the existing map-based system. Therefore processing speed would be one of the key issues that arise. Together with rapid development of the semiconductor industry that results in production of costly gas analyzer, the concept of virtual sensors has been introduced to quantify exhaust emissions to assist engine performance analysis.

Although costly online analysers have been widely used, the emission measurement from a sensor may become interrupted due to instrument failure. A data-driven virtual sensor is an inferential model developed from process observations. Early work on virtual sensor development assumed that a process model was available. Joseph and Brosilow [51] report an inferential model developed using a Kalman filter. In case the process mechanisms are not well understood, empirical models, such as neural network [52] [53] and multivariate statistical methods, are used to derive a regression model [54][55].

2.5.1. Emission modeling and evaluation techniques

There are several conventional methods to predict emissions from IC engine ranging from the simplest ones as the fit algorithm and the engine map to more complicated modeling methods. Engine map method determines the components of exhaust gas based on hard coded data in the form of look up tables; therefore it would be limited to only a certain number of collected data. The fit algorithm is a mean to statistically correlate measured NO_x data collected for a variety of test conditions.

Pitsch et al [56] developed a predictive model for 3-D calculations of diesel engine combustion. This model also described NO_x formation based on detailed chemical reaction kinetics. NO_x emission prediction took into account only the thermal part, described by the Zeldovich mechanism. Due to the fact that estimation of the amount of atomic oxygen relied on

partial equilibrium assumption, NO_x concentration was over predicted. This model claimed to have an error of 4.7% in the estimation.

In research by Dodge et al [57], a diesel engine cycle simulation with detailed calculations of composition for the intake and exhaust gases was used to calculate stoichiometric, adiabatic flame temperatures and expressions that correlate NO_x emission with flame temperatures. This simulation came up with prediction of relative changes in NO_x levels and error was 2% at medium engine load.

Bazari [58] combined a nonlinear transient engine cycle software simulation with a versatile quasi two-dimensional multi-zone combustion emission model in order to predict exhaust emissions under transient operating conditions. The model took governor settings, engine brake load and ambient conditions as inputs. No specific result error was concluded but deviation between measured and estimated NO_x composition was up to 20%.

2.5.2. Neural Network emission prediction techniques

Artificial neural networks are computer-based algorithms designed to mimic the operation of a brain and are the basis of artificial intelligence. Neural network based engine modeling offers the potential for a multi-dimensional, adaptive, learning control system that does not require knowledge of the governing equations for engine performance or the combustion kinetics of emissions formation that a conventional map-based engine model requires [59]. A number of researches around the work have been established to investigate the possibility of applying artificial intelligence technology in performing the non-linear emissions prediction task in IC engine. They have approached the problem in different ways with different models and algorithms.

Conventional Back Propagation neural network models were employed by Henrike C. et al [60] in their researches to predict IC engine emissions. In the work of Henrike C. et al, only one component of a Diesel engine emission gas was investigated, NO_x, which has been claimed to have so many negative effects on human's health. In order to meet the NO_x

emission level enforced by legislation, one of the techniques to reduce the amount of that component was using Selective Catalytic Reduction (SCR) in which a reductant was added to the exhaust gas to catalytically remove the NO_x. The amount of reductant needed to be sprayed into the exhaust gas depends on the amount of NO_x present and the amount of desired NO_x reduction. This project proposed the use of a Back Propagation neural network model to quickly and accurately predict the NO_x concentration formed by a Diesel engine and results from the model could be used to determine the amount of reductant to be added. The proposed models took into account the change of air temperature, the change of air pressure just before the air enters the engine, the rack position of the engine and the engine speed and produced a predictive NO_x concentration in ppm. The number of hidden layers varied up to 2 and the number of hidden nodes varied from 8 to 40. Average absolute error results were benchmarked against the Linear Fit and Engine Map methods and was claimed to be 6.7% comparing with 13.4% for Linear Fit and 17.5% for Engine Map.

In the research by G J Thompson et al [59], a number of neural network models were built to predict output torque and emissions from a modern heavy-duty Diesel engine (Navistar T444E). The network architecture used in this work was a partially recurrent net with inputs including accelerator position, start of injection, fuel injection pulse width, injection control pressure, intake manifold air pressure, engine coolant temperature, engine oil temperature, exhaust gas temperature, intake manifold air temperature and speed. The experiment has shown to be able to predict the continuous torque and exhaust emissions, for the federal heavy-duty engine transient test procedure (FTP) cycle and two random cycles to within 5% of their measured values after only 100 minutes of transient dynamometer training. The author of this paper testified the potential of applying neural network in emissions virtual sensing, on-board diagnostics and engine control strategy optimization.

In another project from the same research group above, Michael L. Traver et al [61] explored the feasibility of using in-cylinder pressure-based variables to predict gaseous exhaust emissions levels from the Navistar T444 direct injection diesel engine through the use of neural networks. The designed electronic testing system read the signal from the pressure transducer and the in-cylinder parameters were calculated at the rate of once per two revolutions of the

engine by DSP boards. The networks were validated offline and the predicted trends were remarkably similar to the collected real data for NO_X and CO_2. The NO_X prediction network used a combination of peak pressure, combustion duration, Indicated Mean Effective Pressure (gross) (IMEPg), Location of Mass Fraction Burned - 50% (LMFB50), and ignition delay in a Ward 1 network with 10 nodes in the hidden layer. The CO_2 prediction network used peak pressure, IMEPg, maximum burn rate, and the ignition delay.

While the researches mentioned above dealt with Diesel engine only, a study conducted by Liu et al [62] on a CNG-Diesel dual fuel engine targeted at using a Radial Basis Function (RBF) neural network to predict the exhaust gas components. The constructed RBF model predicted CO and NO_x for a given set of inputs comprised of speed, amount of natural gas, amount of diesel and injection timing angle. 100 groups of experimental data over the operation conditions from light load and low speed to heavy load and high speed were collected to train the model and it has claimed to achieve the sum-squared error up to 0.15% after about 15000 cycles. This study has proven the applicability of RBF network in the area of emissions prediction in SI engine. However, the model was highly dependent on the number of experimental sample data.

2.6. CONCLUDING REMARKS

This chapter has provided a comprehensive overview of the Spark Ignition (SI) Internal Combustion (IC) engine structure and control strategy as well as current conversion techniques for transforming conventional engines to dual fuel engines. It also highlighted the need of online emission prediction in automotive engineering. Conventional emission modeling techniques and state-of-the-art neural network use in engine emission prediction have been discussed.

Essential knowledge about artificial neural network and particular network models being employed in this research will be provided in the next chapter.

CHAPTER 3

ARTIFICIAL NEURAL NETWORK AND ITS APPLICATION TO VARIOUS NON-LINEAR DYNAMIC PROCESSES

3.1. INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS

The basic definition of Artificial Neural Networks is that it is a type of information processing system whose architecture is inspired by the structure of biological neural systems [63]. These biologically inspired methodologies of computing are thought to be the next major advancement in many technical fields in the future. Even simple animal brains are able to perform functions that are currently impossible for computers.

Artificial Neural Networks are among the newest signal-processing technologies in the engineer's toolbox. They are electronic models based on the neural structure of the brain. The brain basically learns from experience.

An Artificial Neural Network is an adaptive, most often nonlinear system that learns to perform a function (an input/output map) from data [64]. Adaptive means that the system parameters are changed during operation, normally called the *training phase.* After the training phase the ANN parameters are fixed and the system is deployed to solve the problem at hand (the *testing phase).* In general, Neural Networks are non-algorithmic, non-digital and intensely parallel systems consisting of a number of very simple and highly inter-connected processing units that are analogs of the biological neural cells in the brain [65].

3.1.1. Biological Neural Structure

Figure 3.1 depicts the major components of a typical nerve cell in the central nervous system.

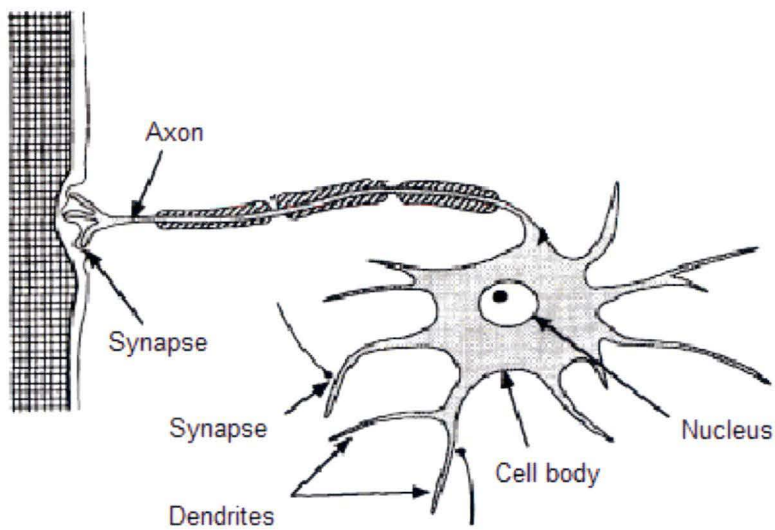


Figure 3.1: *Major structures of a typical nerve cell* [66]

The exact mechanisms of operation of the human brain are still a mystery. Yet, some features of this amazing processor are known. The human brain contains about 10^{11} neurons and for almost all organic neurons one consists of three different parts: a cell body, a set of incoming fibers (the dendrites) and one outgoing fiber (the axon) [67].

The cell body, which is the main processing unit, accepts inputs from other similar neurons, processes these inputs and sends a single output to other neurons in the system. An axon is a long cylindrical connection that carries impulses from the neuron and stretches out to join the dendrites of other neurons through connections called synapses. Synapses release chemical substances that cause a change in the electrical potential of the cell body. When the potential reaches a threshold, an electrical pulse is sent down through the axon which then later reaches synapses and causes them to increase or decrease their potential [68]. Many, but not all, synapses exhibits plasticity; that is, they can increase or decrease in strength under proper conditions and therefore they can have different strengths or synaptic weights [63]. Each neuron can have 10^4 synapses [69].

The learning ability of human beings is probably incorporated in the facility of changing the synaptic weights of those synapses. Donald O. Hebb was among the first that postulated the learning mechanism: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased"[70].

3.1.2. Artificial neuron

Biological neurons and their connections form a system that is asynchronous and not binary. Artificial neural networks try to imitate the complicated and powerful biological neural organism. However, this effort has never reached the state that artificial neural networks can actually replicate human brain. Artificial neural networks are more likely alternative, or event better, methods and technologies to solve technical problems in engineering or computing areas.

In 1943, Warren McCulloch and Walter Pitts proposed the concept of modeling a biological neuron [71]:

" *A synthetic neuron forms a weighted sum of the action potentials which arrive at it (each of these potentials is a numeric value which represents the state of the neuron which has emitted) and then activates itself depending on the value of this weighted sum. If this sum exceeds a certain threshold, the neuron is activated and transmits a response (in the form of an action potential) of which the value is the value of its activation. If the neuron is not activated it transmits nothing.*"

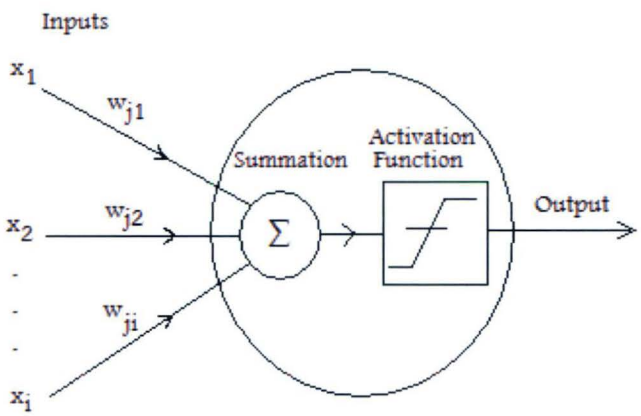


Figure 3.2: *Basic architecture of an Artificial Neuron (Perceptron)*

Figure 3.2 depicts the basic architecture of an Artificial Neuron proposed by Fran Rosenblatt in the early 1960s, illustrating the similarity between artificial and biological neuron. In the layout, various inputs to the network are represented by the mathematical symbol, x_i. Each of these inputs is multiplied by a connection weight that is represented by w_{ji} (j stands for the index of the neuron in the array of neurons and i denotes the index of the input that is connected to this neuron). These products are simply summed, fed through an activation function to generate a result, and then output.

In the simplest case, the process is mathematically illustrated as below:

$$net_j = \sum_{i=1}^{n} x_i w_{ji} \tag{3.1}$$

$$output = f(net_j) = \begin{cases} +1 \ \ if \ \ net_j \geq \theta \\ -1 \ \ if \ \ net_j < \theta \end{cases} \tag{3.2}$$

where

n: the number of inputs

x: input

w: connection weight

f: activation function

θ: neuron's threshold

This simplest type of activation function is called a sign function and this basic form of neural network is called a perceptron.

Besides sign function, there have been many other types of activation functions being used by researchers in Figure 3.3. In general, four functions that are commonly chosen are: the step, sign, linear and sigmoid function.

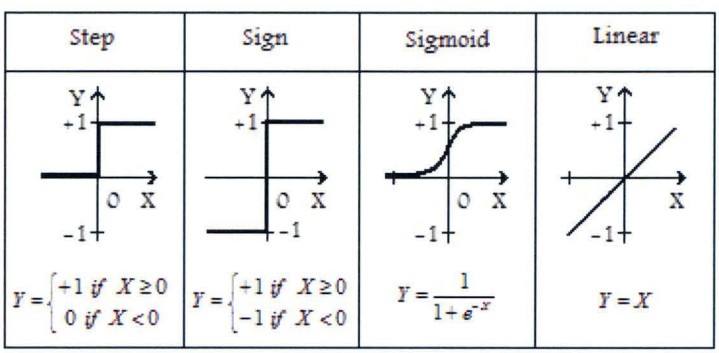


Figure 3.3: *Activation functions*

3.2. NEURAL NETWORK MODELS

The fact that a system with only one perceptron can only cope with problems that are linearly separable has been proven in [68]. However, the invention of perceptron concept has set up an important background and has been inspiration for many other researchers to further develop much more complicated and efficient Neural Network models.

Neural Network models are classified as either feed-forward or recurrent networks, reflecting the way they operate.

3.2.1. Feed-forward Neural Network models

Figure 3.4 shows a basic feed-forward network containing three distinct layers: input layer, hidden layer and output layer. In this topology, the inputs of each neuron are the weighted sum of the outputs from the previous layer. There are weighted connections between the outputs of each layer and the inputs of the next layer. If the weight of a branch is assigned a zero, it is

equivalent to no connection between correspondence nodes. The inputs are connected to each neuron in hidden layer via their correspondence weights. Outputs of the last layer are considered the outputs of the network [72][73]

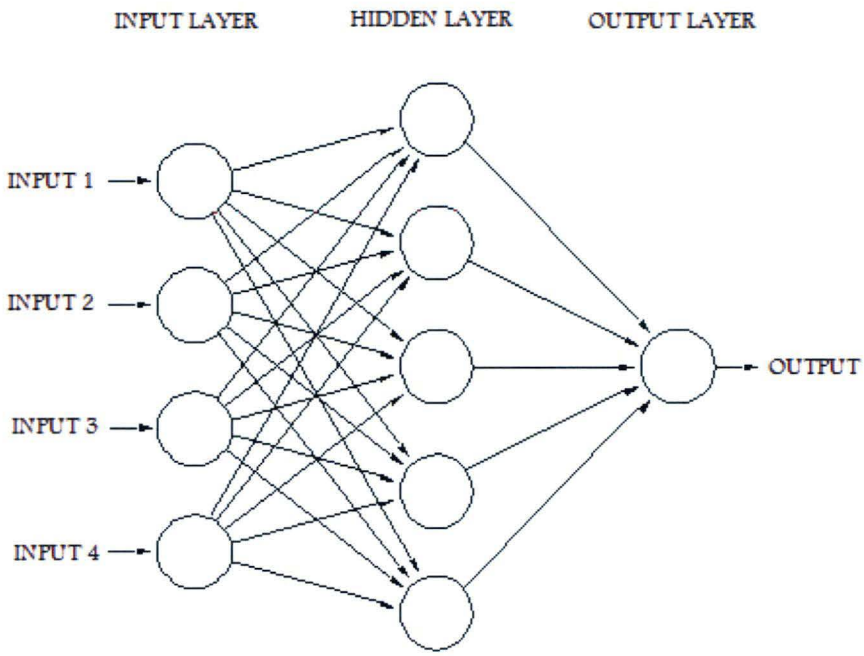


Figure 3.4: *Feed-forward Neural Network*

3.2.2. Recurrent Neural Network models

A recurrent Neural Network differs from a feed-forward Neural Network in the fact that it has at least one feedback loop [74]. As in Figure 3.5, the inputs of the network consist of both external inputs and the network outputs with some delays. Examples of feedback algorithms include the Hopfield network and the Boltzman Machine [75] [76]

Recurrent neural networks exhibit dynamic behaviors because of the integration of feedback loops. They are usually used for caching information as associative memories and solving computationally intensive problems. An important issue with this type of network is the stability and convergence of the model.

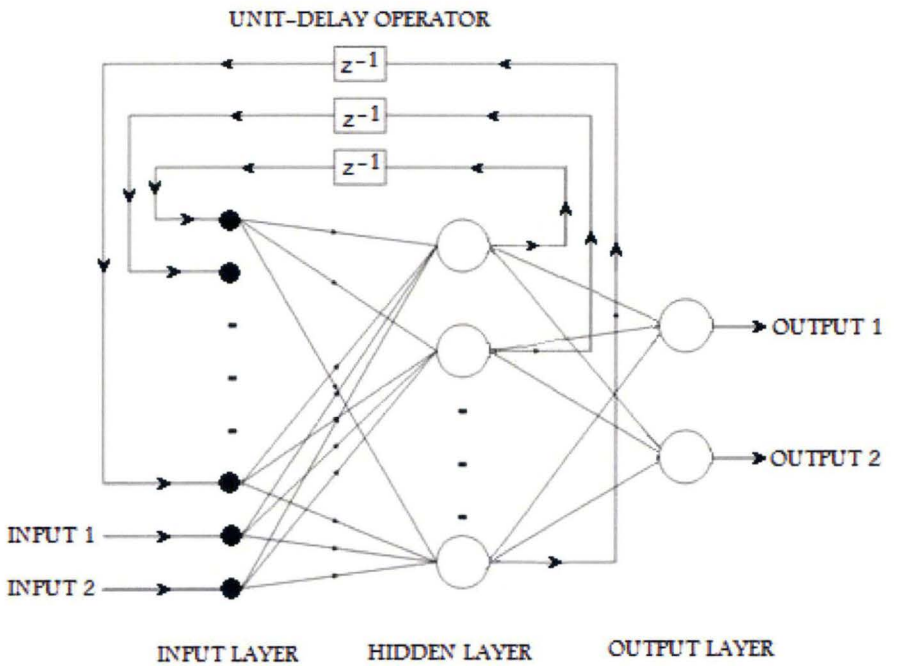


Figure 3.5: *Recurrent Neural Network*

3.3. NORMALISATION OF DATA SETS

Data normalisation is critical in Artificial Neural Network. It makes input and output data have the same order of magnitude. If they have values in different ranges, some variables would seem to have greater influence in the dataset than others if their values vary in larger ranges while that might not be true. The common way of normalising data to the range from 0 to 1 is shown in Equation 3.3 below:

$$Normalised \quad value = \frac{actual \quad value - \min imum \, value}{\max imum \quad value - \min imum \, value} \tag{3.3}$$

3.4. TRAINING OF ARITIFICIAL NEURAL NETWORKS

Neural Networks are not programmed to solve a particular problem; they are trained to do so. The ability to learn is a fundamental feature of intelligence. The network must learn the connection weights from training patterns that are attributed to the network. Performance is hopefully improved over time by iteratively updating the weights in the network. Algorithms for varying the weights so that the training process converges to the desired outcomes are called "learning rules".

The fact that Artificial Neural Networks learn from examples remarkably attracts researchers seeking alternatives to replace conventional computing methods. Instead of following a set of rules specified by human experts, Artificial Neural Networks learn underlying rules from the given collection of representative examples.

There are two types of learning in Neural Network training: supervised training and unsupervised training.

3.4.1. Supervised learning

Most of Artificial Neural Network models are trained with supervision. Supervised learning requires an external teacher to control the learning and incorporates global information. The teacher may be a training set of data or an observer who grades the performance. Examples of supervised learning algorithms are the least mean square (LMS) algorithm and its generalization, known as the back propagation algorithm [78-81] and radial basis function network [82-85].

In this mode, the actual output of a network is compared to the target output. Training consists of presenting input and output data to the network. This data is often referred to as the training set. For each input set provided to the system, the corresponding desired output set is provided as well. Network weights are randomly set at the beginning and are then adjusted so that in the next iteration the model would generate a closer match between the desired and actual output. Adjustment of the weights is toward the direction that minimizes the current errors of all

neurons and is implemented over iterations until an acceptable network accuracy is reached. Once the minimum error is achieved the network weights remain unchanged and the process is said to have converged.

Iteration in the training process is also referred to as "epoch" and the number of input-output patterns in the training set is called "epoch size". Training sets need to be comprehensive and large enough to contain all the features and relationships of the problems to be solved. Therefore, proper selection of the training sets is critical to the performance of the network. [85]

After a supervised network performs well on the training data, the network model needs to be tested with data it has not seen before. This step is important since it ensures that the network has learned the pattern to generate desired results and has not just memorized a given set of data. If this phase does not produce reasonable outcomes the training phase has not been succeeded.

3.4.2. Unsupervised training

In contrast to supervised learning, unsupervised learning does not require external teachers. The system the system must organize itself by internal criteria and local information designed into the network. Unsupervised learning is sometimes referred to as self-organizing learning. Kohonen is one of the networks that use unsupervised learning [68].

Unsupervised training only requires input vectors to train the network. During the learning process, it discovers significant features in the input patterns and learns how to classify them into appropriate categories. In other words, this type of network uses no external influences to adjust their weights but internally monitor the performance. It extracts regularities or trends in the input patterns and makes adaptations according to the function of the network.

3.5. NEURAL NETWORK SELECTION

Neural Networks have provided an alternative way to analyse data and have their effectiveness proven in solving a broad range of computing problems. However, they are not the solution for all problems. Selection of proper models is critical when one decides to employ the biologically inspired technology in their applications since wrongly chosen methods would result in significant loss of resources and accomplishing time would be delayed.

Although contemporary application of Neural Networks is extremely broad, this chapter only investigates a limited number of Neural Networks that are utilised in 2 scenarios:

- Data classification
- Prediction

Since the scope of this research includes using Artificial Intelligence techniques for engine emission prediction, the Neural Network targeted to be used for that purpose would be described in great detail in the following section

3.5.1. Data classification

A Kohonen network is also referred to as self-organising feature map or topology preserving map, reflecting the property of capturing an important aspect of the feature maps in the cortex of highly developed animal brains [86]. It transforms an incoming signal pattern of arbitrary dimension into a one or two-dimensional discrete map adaptively in a topological ordered fashion [87].

Kohonen neural network uses competition in training. A neuron learns by shifting its weight from inactive connections to active ones. Only the winning neuron and its neighbour neurons are allowed to learn. If a neuron does not respond to a given input pattern, learning does not take place in that particular neuron. [68]

For every input pattern presented to the network model, a winning neuron is determined; its weight as well as those of the neurons in its proximity are updated. Going through a number of iterations, the network weights will converge to a specific weight matrix.

The key difference between this type of network and conventional supervised training models is that a numerical measure of the magnitude of the mapping error is not possible. Nevertheless, the training process results in the determination of well-defined parameters for a given application [88].

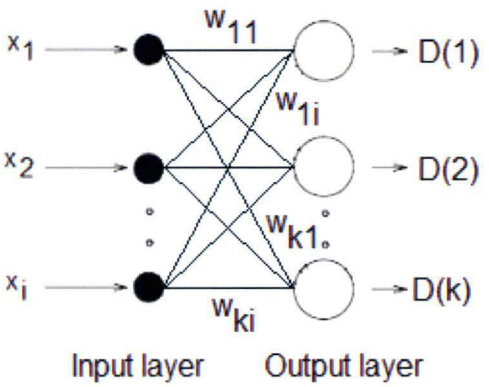


Figure 3.6: *Basic Kohonen Network*

The structure of a basic Kohonen neural network is shown in Figure 3.6. For the outputs from the network, the output neuron that has the minimum value is the winning neuron. Only the weight of that neuron is allowed to be adjusted.

The general training algorithm of Kohonen network is listed as follows [89]:

Step 1: Initialise network weights w_{ki}, set topological neighbourhood parameter R, set learning parameter η.
Step 2: While stopping condition is false, proceed to step 3.
Step 3: For each input vector x, do steps 4 to 9.
Step 4: For each output neuron k, calculate:

$$D(k) = \sum_i (w_{ki} - x_i)^2 \qquad (3.4)$$

Step 5: Find index k such that D(k) is a minimum.

Step 6: For all output units k within a specified neibourhood of K, and for all inputs i, calculate:

$$W_{ki}(new) = w_{ki}(old) + \alpha[x_i - w_{ki}(old)] \qquad (3.5)$$

Step 7: Update the learning rate and topological neighbourhood parameters.

Step 8: Reduce radius of topological neighbourhood at specified times.

Step 9: Test stopping condition.

The accuracy of the map depends on the number of iteration of the self-organising feature map algorithm, the selection of main parameters of the algorithm such as the learning rate η and the topological neighbourhood parameter R [89],

The learning rate parameter is time varying and should initially start at a value close to 1 and then decrease gradually with increasing iterations to a value above 0.1.

Initially, the topological neighbourhood parameter R should be set to a value such that it includes all neurons in the network, then gradually decreases over the period of iterations such that it encompasses only the winning neuron and may be a single neighbour as the training coming towards the final iterations.

3.5.2. Prediction

3.5.2.1. Back-propagation Neural Network

A Back-propagation Neural Network is a multi-layered feed-forward neural network. It is trained with the supervised training method called the error back-propagation procedure [90]. Figure 3.7 depicts a simple Back-propagation Neural Network model with an input layer having i number of inputs, one hidden layer with j neurons and one output layer with k outputs. In this pattern the inputs are passed forward to the output layer via the hidden layer.

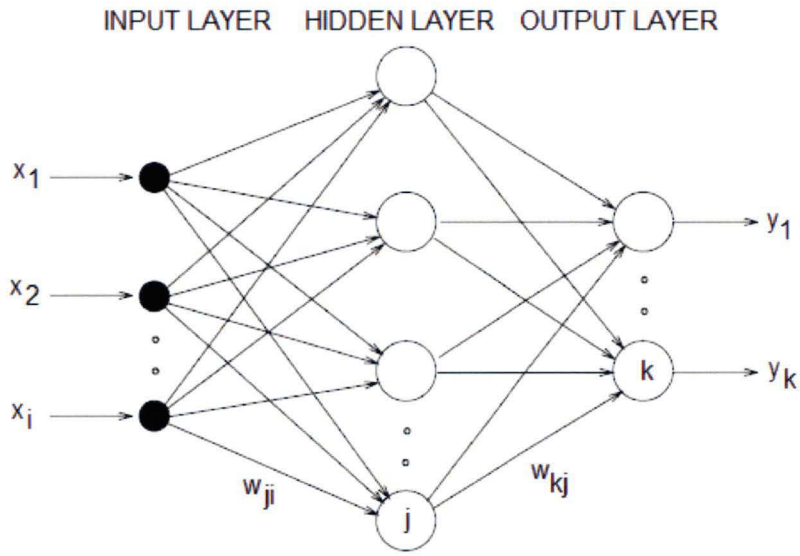


Figure 3.7: *Back-propagation neural network model with one hidden layer*

Weight of a connection within the network is denoted by the letter w. The first index of the weight indicates the index of target neuron and the second index is the index of the source neuron. In this type of network, every neuron in each layer communicates only with neurons in the immediately following layer and only neurons in the hidden layer and the output layer perform processing functions. The neurons in those layers perform three functions: an input function, an activation function and an output function.

The input function is given by the sum of products of inputs and their corresponding weights as shown in equation (3.6):

$$net_j = \sum_j x_i w_{ji} \qquad (3.6)$$

where

net_j = sum of weighted input to neuron j

x_i = input i to neuron j

w_{ji} = weight of connection from neuron i to neuron j

The activation function usually used in Back-propagation neural network is the non-linear sigmoid function:

$$f(net) = \frac{1}{1+e^{(-net)}} \qquad (3.7)$$

The output function simply passes the output of the activation function forward to the connected neuron in the most adjacent layer in the direction of the information flow.

Back propagation algorithm is one of the most popular algorithms for training a network due to its success from both simplicity and applicability viewpoints. Training of a Back-propagation Neural Network is implemented in two separate stages. Firstly, the input data set is presented to the network and generates a forward flow of activation from the input to the output layer. Secondly, network output errors generate a flow of information from the output layer backward to the input layer

The error Back-propagation training uses a gradient descent method that adjusts the weight by making a change in the weight by an amount proportional to the partial derivative of the error function with respect to the given weight [68]. In this training procedure, the error is calculated at the output layer and hidden layers. The calculation is detailed as below:

- For neuron k in the in the output layer, the error value δ_k is given by

$$\delta_k = (t_k - y_k)\, f'(net_k) \qquad (3.8)$$

where

t_k : target output of neuron k

y_k = real output of neuron k

$f'(net_k)$ = derivative of the activation function

net_k = sum of weighted inputs to neuron k

- For neuron j in the hidden layer as shown in the Figure 3.8.

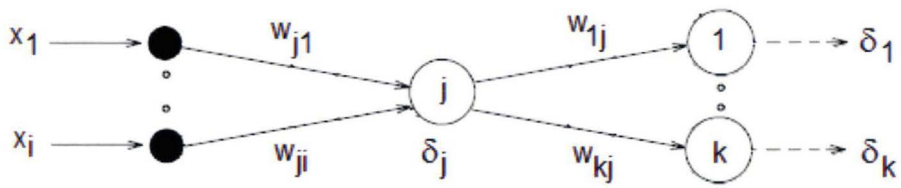


Figure 3.8: *Neuron j in the hidden layer*

Error δ_j of neuron j in the hidden layer is calculated by

$$\delta_j = \left[\sum_k \delta_k w_{kj}\right] f'(net_j) \qquad (3.9)$$

where, w_{kj} is the weight of connection from neuron j to neuron k

Each interconnection weight is adjusted by considering the error term of the neuron that receives input from that connection. The adjustment is performed as below [91]:

$$\Delta w_{ji} = \eta\, \delta_j\, x_i \qquad (3.10)$$

where

w_{ji} = weight of connection from neuron i to neuron j

η = learning rate constant, $0<\eta<1$

x_i = input i to neuron j

Adjustment of weights is primarily based on three parameters: w_{ji} ,η and x_i. The learning rate in the weight updating process reflects the convergence speed of the neural network. Large values of the learning rate would result in instability in the network while very small values would lead to slow training process. In order to cope with the issue, the learning rate is sometimes varied during network training. Starting the first iterations with high value of η and decreasing its value during training might produce better performance [91].

To improve the convergence of the weight training session, an additional term called momentum is proposed and is incorporated in the weight adjustment formula [92]

$$\Delta w_{ji}^{n} = \eta\, \delta_j x_i + \alpha \Delta w_{ji}^{n-1} \qquad (3.11)$$

where

α = momentum constant ($0 \le \alpha < 1$)

Δw_{ji}^{n} = weight correction at iteration n^{th}

Δw_{ji}^{n-1} = weight correction at iteration $(n-1)^{th}$

It can be seen from the formula that a portion of the weight adjustment is not applied until the next iteration. Therefore the effect of oscillation in weight changes is dampened and convergence is improved.

To evaluate the success and sufficiency of the training process, Root-Mean-Squared (RMS) error is commonly used as a measure and is calculated using the following formula [91]

$$RMS\ \ error = \sqrt{\frac{\sum_{p}\sum_{k}\left(t_{kp} - y_{kp}\right)^2}{PK}} \qquad (3.12)$$

where

P = number of training data

K = number of output neurons

t_{kp} = target output for the output neuron k after presentation of the training data p

y_{kp} = output value produced by the output neuron k after presentation of the training data p

Generally a RMS error value less than 0.1 is satisfactory for a training session.

Despite the apparent success of the back-propagation learning algorithm, there are some aspects that make the algorithm not guaranteed to be universally useful. Most troublesome is the long training process. This can be a result of a non-optimum learning rate and momentum. Outright training failures generally arise from two sources: network paralysis and local minima [93].

The general training algorithm is summarized below [86]:

Step 1: Initialise the weights of the network at small random values.

Step 2: Start the learning cycle by exposing the network to a certain input pattern paired with the desired output.

Step 3: Compute the network output (Equation 3.6 and 3.7). Compare that with the desired output and calculate the error for the output layer and the hidden layer (Equation 3.8 and 3.9).

Step 4: Adjust the weights of the network using the error Back-propagation algorithm (Equation 3.10 or 3.11)

Step 5: Repeat steps 2 to 4 with all the input patterns and their desired outputs. Compute the cumulative error (Equation 3.12)

Step 6: If the cumulative error is not within the desired range, go back to step 2. Otherwise, terminate the training process.

3.5.2.3. Radial Basis Function Neural Network

Radial Basis Function (RBF) Neural Network is a special type of multi-layer feed-forward networks in which each unit in the hidden layer employs a radial basis function, such as a Gaussian kernel, as the activation function [87] shown in Figure in Figure 3.9 below.

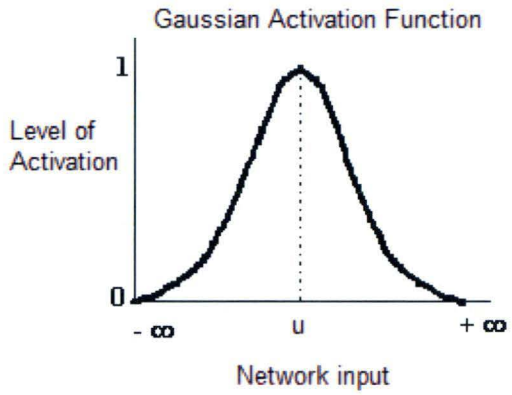


Figure 3.9: *Gaussian function*

The Gaussian activation function has exponential form and is represented by the following formula:

$$h_j = \exp\left[-\frac{(x-u_j)^T(x-u_j)}{2\sigma_j^2}\right] \tag{3.13}$$

where

h_j = output of hidden layer neuron j

x = input vector

u_j = weight vector of hidden layer neuron j

T = vector transpose operator

σ_j = diameter of receptive field of hidden layer neuron j

The radial basis function (or kernel function) is centered at the point specified by the weight vector associated with each unit in the hidden layer. The widths and positions of these kernels must be learned from training patterns.

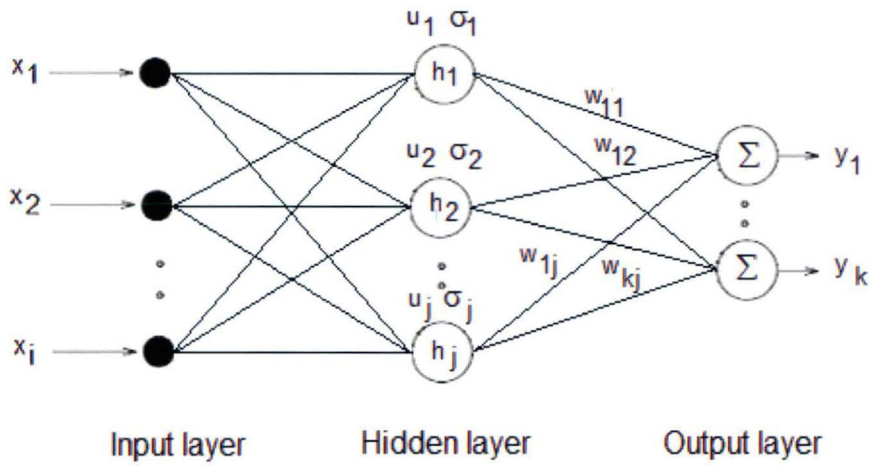


Figure 3.10: Basic Radial Basis Function Neural Network

Basic structure of a Radial Basis Function Neural Network is shown in Figure 3.10. Output y of an output neuron is calculated by:

$$y = \sum_j h_j w_{kj} \tag{3.14}$$

where

h_j = output of hidden layer neuron j

w_{kj} = output layer weight

Training of the radial basis network includes two stages. In the first stage, the center u and diameter of receptive σ of each neuron will be assigned. At the second stage of the training, the weight vector W will be adjusted accordingly. After the training phase is completed, the next step is the recall phase in which the outputs are applied and the actual outputs of the network are produced.

Finding the center u of each neuron

One of the most popular approaches to locate the centers u is to divide the input vector to some clusters and then find the center of each cluster and locate a hidden layer neuron at that point [95]. In this step, a Kohonen network can be used to cluster the training patterns to reduce the number of function centers. Such configuration is called Radial Basis Function incorporating Kohonen network.

Finding the diameter σ of the receptive region

The value of σ can have significant effect on the performance of the network. There are different approaches to find this value. One of the popular methods is based on the similarity of the clustering of the input data. For each hidden layer neuron, the RMS distance of each neuron and its first nearest neighbor will be calculated; this value is considered as σ [89].

The training algorithm is summarised below [94]:

Step 1: Use a suitable clustering technique to set the input-to-hidden layer weights of the network to represent sufficiently the training patterns. Initialise the hidden-to-output layer weights of the network at small random values.

Step 2: Start the learning cycle by exposing the network to a certain input pattern paired with the desired output.

Step 3:Comput the network outputs using Equation 3.13 and 3.14. Compare with desired outputs to come up with the error term.

Step 4:Adjust the hidden-to-output layer weights using the error Back-propagation algorithm. The input-to-hidden layer weights remain unchanged.

Step 5: Repeat steps 2 and 4 with all the input patterns and their corresponding desired outputs. Compute the cumulative error using Equation 3.12

Step 6: If the cumulative error is not within the desired range, go back to step 2. Otherwise, terminate the training session.

3.5.2.4. Optimisation Layer by Layer Neural Network

This type of network is introduced by Ergezinger and Thomsen [96]. The feed-forward learning process is accelerated since the weights during training is optimised layer-by-layer. It is declared to have reduced the lengthy time the Back-propagation algorithm would take for the network to converge.

The Optimisation Layer by Layer Network reduces the optimisation of connection weights in each layer to a linear problem, which can then be solved exactly [96]. Nevertheless, to reduce the number of unavoidable linearisation errors, a penalty term is introduced to the cost function so that layers are optimised alternately in an iteration process. This network category contains no parameter that can be interacted with by users [96].

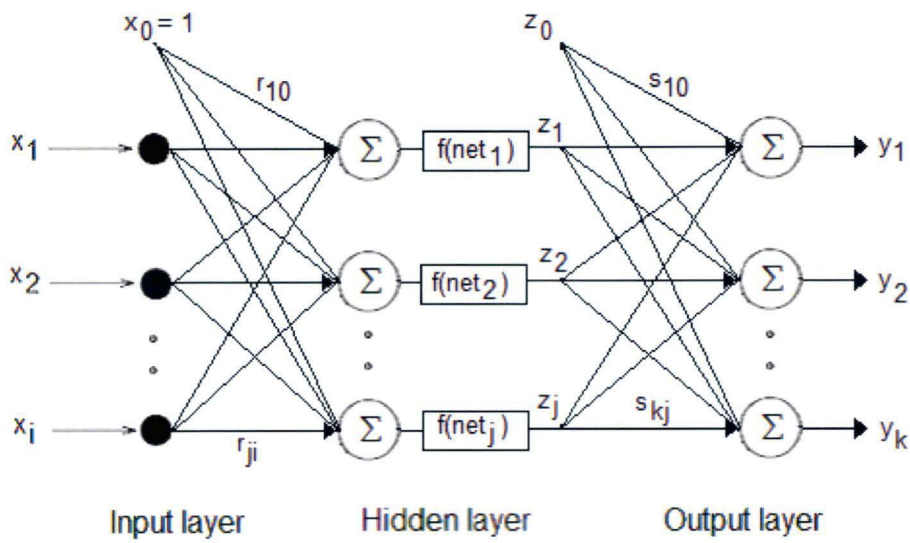


Figure 3.11: *Basic Optimisation Layer by Layer Neural Network*
(with i=0..I;j=0..J;k=0..K)

The basic structure of an Optimisation Layer by Layer Neural Network (OLLNN) is shown in Figure 3.11. Input neurons are connected to the hidden layer via connections with weights r_{ji} and hidden neurons are connected to output neurons through weighted connections s_{kj}.

Unlike the Back-propagation network, the OLLNN only uses the non-linear sigmoid function as activation function in the hidden layer. The output neurons use linear activation function.

Activation function for hidden neuron

$$f(net) = \frac{1}{1 + \exp(-net)} \qquad (3.15)$$

Activation function for output neuron

$$f(net) = net \qquad (3.16)$$

where

net = sum of weighted inputs to neuron as shown in Equation 3.6

Basic features of an OLLNN are [96]:

- The weights in each layer are modified dependent on each other, but separately from all other layers.
- The optimisation of weights in the output layer is a linear problem.
- The otimisation of the weights of the hidden layer is reduced to a linear problem by linearising the sigmoid activation function.

The weight is adjusted towards minimising the cost function, which is the presentation of the error between desired and real network outputs.

The cost function is given by:

$$E(R,S) = \frac{1}{P}\sum_{p}^{P}\frac{1}{2}\left(t^p - y^p\right)^2 \qquad (3.17)$$

where

t^p = desired network output of training data p

y^p = network output of the training data p

P = number of training data

R = Weight matrix of hidden-input connections

S = Weight matrix of output-hidden connections

Adjustment of the weight matrix is required for the output layer and the hidden layer

Adjustment of the output layer weight matrix

Firstly, the gradient of the cost function E with respect to s needs to be calculated and set to 0

$$\frac{dE}{ds} = \frac{1}{P}\sum_{p=1}^{P}\left(s^T z^p - d^p\right)z^p = 0 \qquad (3.18)$$

where

$s^T z^p$ = equivalent to the network output y

z^p = the scalar output of the hidden neuron of training data p

d^p = desired output of training data p

The optimal output layer weight matrix S^{opt} can be found by solving the Equation 3.19 and 3.20 below [96]:

$$A.S = b \qquad (3.19)$$

$$S^{opt} = A^{-1}.b \qquad (3.20)$$

where

$$A = \text{matrix}[a_{h,j}]\ ;\ a_{h,J} = \sum_{p=1}^{P} z_h^p z_j^p \qquad h,j = 0..J$$

$$b = \text{matrix}[b_{k,j}];\ b_{k,J} = \sum_{p=1}^{P} t_k^p z_j^p \qquad j = 0..J;\ k=1..K$$

Since the weight adjustment procedure is straightforward and iterative process as in the Back-propagation network is avoided the network training time is reduced.

Adjustment of the hidden layer weight matrix

Calculation for the optimal hidden weight matrix R^{opt} is similar to the one presented above. However, the non-linear sigmoid activation functions are required to be transformed into a set of linear equations by using Taylor series expansion. The weights of connections between the hidden layer and the output layer are linearised (s*lin*) and calculated by the following equation:

$$slin_j = f'(net_j).s_j \qquad (3.21)$$

where

$f'(net_j)$ = derivative of summation of weighted inputs to neuron j

s_j = weight of connection from neuron j to a neuron in the output layer

The resulting linearised network structure depicting the linear relation between the output change Δy_k of any neuron k and the weight change Δr_{ji} is shown in Figure 3.12. Optimisation of the hidden layer weight matrix is based on this structure.

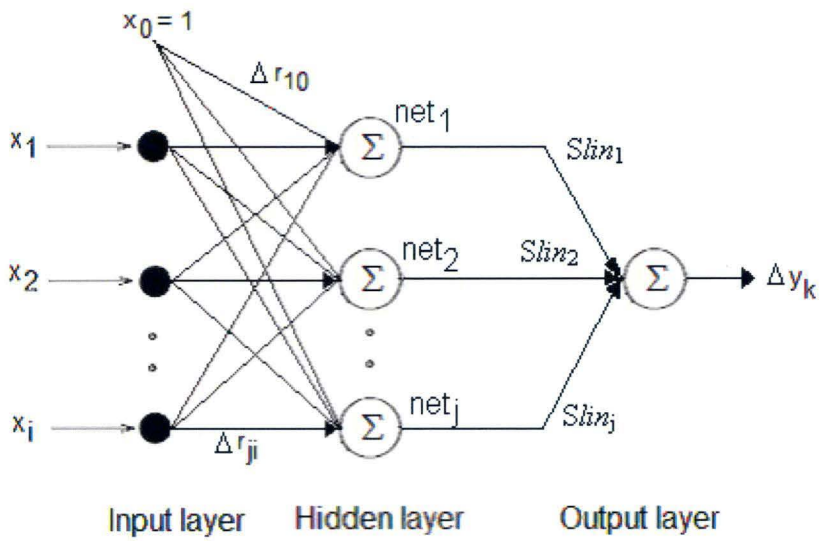


Figure 3.12: *Linearised Network for the hidden layer connections at neuron k*

Since the hidden layer weights have undergone through the linearization process, the cost function for the hidden layer is given by:

$$E_{hidden} = E_{linear} + \mu Epen \tag{3.22}$$

where

E_{hidden} = overall error function for the hidden layer

E_{linear} = error function for the linearised sigmoid function

E_{pen} = penalty term to account for linearisation error

μ = penalty constant that determines the influence of the penalty term on the linear cost function

The optimal weight change matrix is for the hidden layer connection is calculated by:

$$\Delta R^{opt} = \tilde{A}^{-1} . \tilde{b} \tag{3.23}$$

where

$$\tilde{A} = matrix[a_{jl,hm}]$$

for j≠h, $a_{(jl,hm)} = \sum_{p=1}^{P}\sum_{k=1}^{K}\left[(slin_{kj}^{p}.x_{i}^{p})(slin_{kh}^{p}).x_{m}^{p}\right]$ where i,m=0..I;j,h=0..J;k=1..K

for j=h, $a_{(jl,hm)} = \sum_{p=1}^{P}\sum_{k=1}^{K}\left[(slin_{kj}^{p}.x_{i}^{p})(slin_{kh}^{p})x_{m}^{p} + \frac{\mu}{J}\left|s_{kj}\right|.\left|f''(net_{j}^{p})\right|.x_{i}^{p}x_{m}^{p}\right]$

$\tilde{b} = vector[b_{jl}]$ where $b_{(jl)} = \sum_{p=1}^{P}\sum_{k=1}^{K}\left[(t_{k}^{p} - y_{k}^{p})(slin_{kh}^{p}).x_{m}^{p}\right]$

$slin_{kj}^{p}, slin_{kh}^{p}$ = linearised weight from neuron k at the output layer to the hidden neuron j,h (of the training data p)

x_{i}^{p}, x_{m}^{p} = input of the neuron i,m at the input layer (of the training data p)

s_{k}^{j} = weight of connection from output neuron k to hidden neuron j

$f''(net_{j}^{p})$ = second derivative of the sigmoid function of net_j

Once ΔR^{opt} is calculated the hidden layer weight matrix is updated by:

$$R_{new} = R_{old} + \Delta R^{opt} \qquad (3.24)$$

The overall training algorithm is described below [96]:

Step 1: Initialise the network weight matrix (R,S) at small random values and set the initial value for the penalty constant (μ=0.0001 is recommended).

Step 2: Start the training cycle by exposing the network to training data.

Step 3: Compute the optimal output layer weight matrix S^{opt} from Equation 3.20. Update the output layer weight matrix $S = S^{opt}$ and calculate the RMS error.

Step 4: Calculate the optimal weight change ΔR^{opt} for the hidden layer using Equation 3.23 and then compute R_{test} from the equation below

$$R_{test} = R_{old} + \Delta R^{opt} \qquad (3.25)$$

Step 5: Calculate a new RMS error from the weight matrix R_{test} and the current weight matrix S or S^{opt}. The new RMS error presented by RMS_{test} is compared with the current RMS error calculated in step 3.

If $RMS_{test} < RMS_{current}$

Update the hidden layer weight matrix $R=R_{test}$ and decrease the penalty constant using:

$$\mu_{new} = \mu_{old} . \beta \qquad (3.26)$$

where $0<\beta<1$ ($\beta = 0.9$ is recommended)

Proceed to step 6

If $RMS_{test} \geq RMS_{current}$

Increase the penalty constant using

$$\mu_{new} = \mu_{old} . \gamma \qquad (3.27)$$

where $\gamma > 1$ ($\gamma = 1.2$ is recommended)

Go back to step 4.

Step 6: If the RMS is not within the desired range, go back to step 2. Otherwise finish the training process

3.5.2.5. Hybrid Neural Network

Hybrid Neural Network is developed at the University of Tasmania by Kiatcharoenpo, T. Karri, V in an attempt to improve the performance of the Optimisation Layer by Layer Neural Network [97].

The basic idea of this type of network is to combine the iterative approach from the Back-propagation algorithm with a direct weight optimisation approach from the Optimisation Layer by Layer algorithm to reduce the training time and to avoid the problem of local minima.

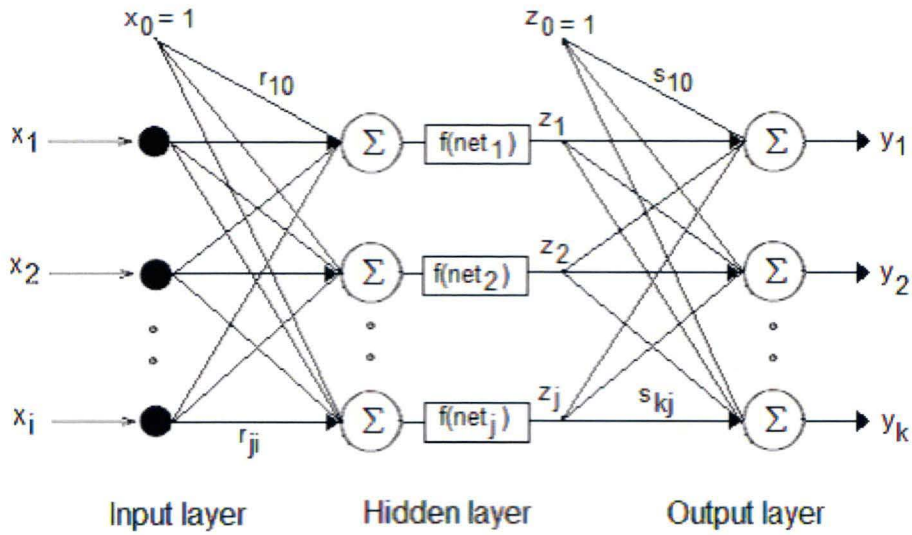


Figure 3.13: *Basic Hybrid Neural Network*

(with i=0..I;j=0..J;k=0..K)

The basic idea of this type of network is to combine the iterative approach from the Back-propagation algorithm with a direct weight optimisation approach from the Optimisation Layer by Layer algorithm to reduce the training time and to avoid the problem of local minima.

Structure of a basic Hybrid Neural Network is shown in Figure 3.13. The architecture of a Hybrid Network is similar to a Back-propagation Network except the fact that its output neurons uses linear activation function.

Outputs of hidden neurons are given by:

$$net_j = \sum_j x_i r_{ji} \tag{3.28}$$

$$z_j = f(net_j) = \frac{1}{1+\exp(-net_j)} \tag{3.29}$$

where

net_j = weighted summed input to neuron j

x_i = input i to neuron j

r_{ji} = weight connecting input neuron i to hidden layer neuron j

$z_j = f(net_j)$ =output of hidden neuron j.

In the output layer, output of each neuron is simply the sum of all of its inputs. The network is trained by initialising small random weights and then adjusting the weights iteratively using the Back-propagation algorithm with some initial iterations to provide initialised weights for further training. The direct weight optimisation method based on the Optimisation Layer by Layer algorithm is then used to finales the weights at a rapidly converging rate [97].

Because of the difference in network architecture between the Back-propagation Network and the Hybrid Network, a new Back-propagation desired output corresponding to the Back-propagation algorithm needs to be generated using Equation 3.30 to train the network.

$$d_k = f(t_k) = \frac{1}{1+\exp(-t_k)} \qquad (3.30)$$

where

d_k = Back-propagation desired output of output neuron k

t_k = desired output at neuron k of the output layer

The new Back-propagation training pattern $(x_0,..,x_i;d_1,..,d_k)$ can then be used. In training of the Hybrid Network an iterative weight adjustment using the Back-propagation algorithm is performed to set appropriate initial weights before a direct weight adjustment using the Optimisation Layer by Layer algorithm is implemented to finalise optimal weights.

The algorithm of Hybrid Network training is detailed below [97]:

Step 1: Initialise the network weight matrix (R,S) at small random values.

Set initial value for a penalty constant (μ=0.0001).

Set the number of initial iterations for training with Back-propagation algorithm

Step 2: Compute Back-propagation training patterns using Equation 3.30.

Step 3: Start initial weight adjustment process by exposing the network to the Back-propagation training pattern generated by step 2. Using the Back-propagation

algorithm, the weights are modified iteratively until the number of initial iterations is complete.

Step 4: Start direct weight optimisation approach using Optimisation Layer by Layer algorithm.

Compute the optimal output layer weight matrix S^{opt} from Equation 3.20.

Update the output layer weight matrix $S = S^{opt}$ and calculate RMS error.

Step 5: Calculate the optimal weight change ΔR^{opt} for hidden layer using Equation 3.23 and compute R_{test} from Equation 3.25.

Step 6: Recalculate RMS error from the weight matrix R_{test} and the current weight matrix S or S^{opt}. The new RMS error presented by RMS_{test} is compared with the current RMS error calculated in step 4.

If $RMS_{test} < RMS_{current}$

Update the hidden layer weight matrix $R=R_{test}$ and decrease the penalty constant using:

$$\mu_{new} = \mu_{old} \cdot \beta \qquad (3.31)$$

where $0<\beta<1$ ($\beta = 0.9$ is recommended)

Proceed to step 7

If $RMS_{test} \geq RMS_{current}$

Increase the penalty constant using

$$\mu_{new} = \mu_{old} \ \gamma \qquad (3.32)$$

where $\gamma > 1$ ($\gamma = 1.2$ is recommended)

Go back to step 5.

Step 7: Terminate the training if RMS error is within the desired range or the number of iterations is complete. Otherwise go back to step 4.

3.6. APPLICATION OF ARTIFICIAL NEURAL NETWORKS

Neural network is one of the promising fields that attract the interest of many researchers around the world. While many of them focus on developing network models with higher accuracy or better training algorithm, many others are exploring the option of integrating neural network into various current technologies.

The range of possible fields that neural network can be applied is broad and might include: character recognition, signal processing and financial...

The first neural network being applied to solve a real-world problem is Multiple ADAptive LINear Elements (MADALINE) developed in 1959 by Bernard Widrow and Marcian Hoff of Stanford [98]. It acts as an adaptive filter to eliminate echoes on phone lines and is incorporated into special purpose chips.

Nowadays, neural networks are making big invasion into the financial worlds. Banking, credit card companies, and lending institutions typically deal with decisions that are not obvious and require experienced staffs and knowledge of statistical trends. Therefore, artificial neural network technology, which is claimed to have the capability of inferring trends and solving for answers based on learning, is the promising candidate.

Neural networks can be used from the first stage of the financial process such as filling application forms. In the commercial products developed for financial customers by Fair Isaac Corporation, a neural-network-based Optical Character Recognition (OCR) system is incorporated to recognize hand printed characters through a scanner. OCR can take cards, like a credit card application form, and put those recognized characters into a database [99]. Furthermore, neural networks can also assist officers from credit card companies to establish credit risks and credit limits of credit card applicants. Based on the result of analysis, grant of card can be accepted or declined.

Within the scope of this research, another use of neural networks in the automotive industry is explored: the capability of artificial intelligence in engine emission prediction. As reviewed in Section 2.5.2, there have been a number of studies investigating the potential of using artificial neural technology to predict emissions from engine exhaust and have achieved promising results.

3.7. CONCLUDING REMARKS

This chapter has provided a general knowledge of artificial neural network its associated concepts. It has also given a review of the neural network models that are going to be used through out this research in details. Four types of models have been mentioned: Back-propagation, Radial Basis Function, Optimisation Layer by Layer and Hybrid Neural Network. Among those categories, the last two types are fast-learning networks .

This chapter has also given information about applying neural network technologies into solving real-world problems. The fields in which integration of neural networks is possible are character recognition, signal processing and financial... and there can be a lot more.

In the next chapter, design of an experimental test rig for exploration of neural network capability in predicting emissions from a dual fuel Hydgrogen-Gasoline engine will be described. The neural networks examined above are targeted to be incorporated into the test rig to perform the prediction task and act as a virtual emission sensor (software sensor).

CHAPTER 4

EXPERIMENTAL TEST RIG SET-UP AND CALIBRATION PROCEDURES

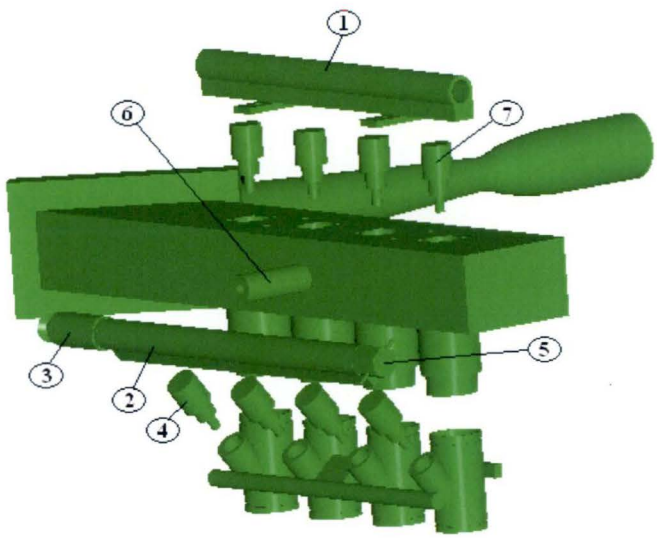
4.1. INTRODUCTION

In development of an emission virtual sensor, extensive testing is required to prove the applicability and practicality of the model. Moreover, to assist the learning stage of the artificial neural network, a test-bed that provides a large range of engine dynamic is required. This chapter describes the experimental set-up that is used to obtain training data for neural network models and to appraise those models. This chapter provides a description of:

- Design of the dual-fuel intake manifold
- Control system of the engine
- Associated sensors
- Initial tuning procedure of the Engine Management System

4.2. DUAL FUEL INJECTION MANIFOLD DESIGN

To allow mixing of Hydrogen and Petrol, a dual-fuel injection intake mechanical model has been developed on CADKEY98 (3D modeling software package). The real manifold was built by modifying the intake manifold originally designed for the University's Formulae SAE racecar running on Petrol in 2001/2002 as shown in Figure 4.1.



1-Petrol fuel rail
2-Hydrogen fuel rail
3-Flashback arrestor
4-Hydrogen injector
5-Hydrogen relief valve
6-Adjustable pressure relief valve
7-Petrol injector

Figure 4.1: *3D model of the dual-fuel intake manifold assembly*

This design of the dual-fuel assembly is a single plane induction manifold with intake runners extending from a common square plenum manufactured out of 5mm aluminum plate with an approximate volume of 3L. Fuel rails for Hydrogen and Petrol are also made of aluminum. The

plenum is fitted with an adjustable pressure relief valve that was set at 3-bar pressure. It functions as a safety valve that helps release pressure inside the plenum if Hydrogen gets accumulated inside and explodes.

Before supplying Hydrogen to the Hydrogen fuel rail, the rail needs to be purged with nitrogen to ensure no air is left inside. If air remains in the rail with Hydrogen, explosion might occur. The relief valve on the rail is opened and closed a number of times when purging the fuel rail to release nitrogen and air inside (if exists).

To avoid explosion in the main Hydrogen supply when backfiring occurs and ignites the gas inside the rail, a flashback arrestor is fitted on the other end of the Hydrogen fuel rail. Four Petrol injectors are controlled by the MoTeC Engine Management System and four others Hydrogen fuel injectors are controlled by the embedded add-on injection system as shown in Figure 4.2

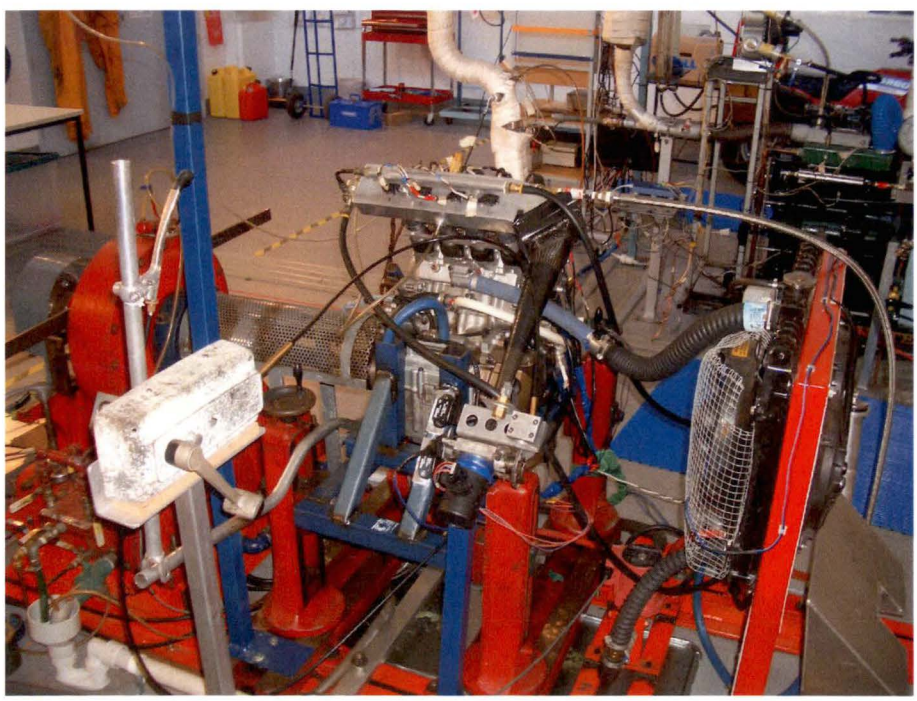


Figure 4.2: *Experimental test rig set-up*

4.3. CONTROL SYSTEM

Control system of the dual fuel engine is comprised of a MoTeC M4 Engine Management System and a custom-built add-on fuel injection system. The add-on system includes a PC104 computer and a Motorola HC12 micro-controller.

The MoTeC M4 Engine Management System shown in Figure 4.3 is a 32-bit engine control unit that provides sequential injection and different mode of ignition control (wasted spark ignition and sequential spark ignition) with programmable look-up tables.

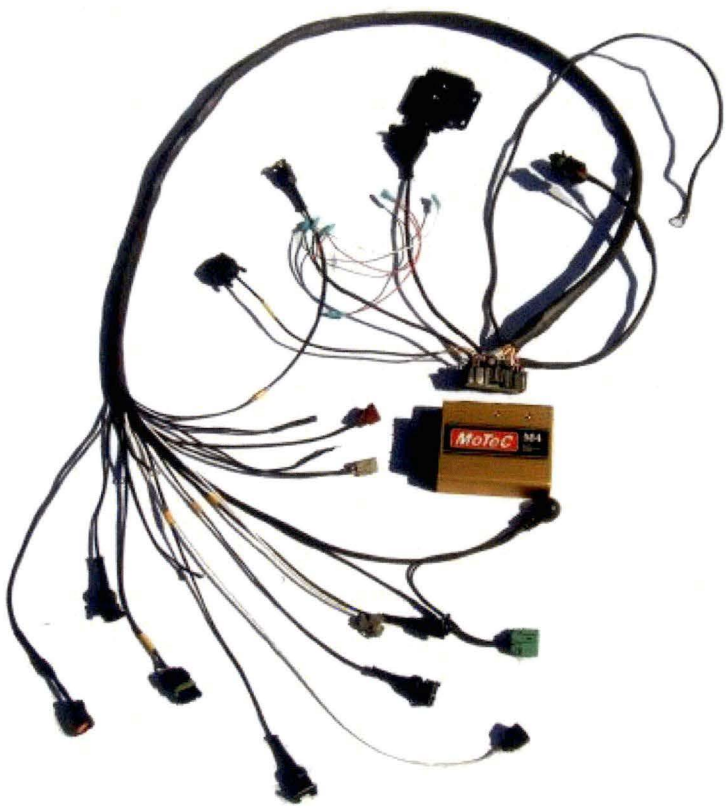


Figure 4.3: *MoTeC M4 Engine Management System*

Essential inputs and outputs of the MoTeC system are shown in Figure 4.4. Pretty much like other commercially available systems in the market the MoTeC unit determines the amount of

fuel to be injected and the ignition advance angle by calculating engine speed and load then extracting the corresponding injection and ignition data from appropriate look-up tables.

There are two types of sensors that can be used to characterise engine load. They are: manifold air pressure sensor and throttle position sensor. In this set-up the throttle position sensor type has been chosen as the indicator of engine load. With this arrangement for dual fuel operation the MoTeC Engine Management System is responsible for timing 4 Petrol injectors positioned on the top of the manifold assembly as shown in Figure 4.5 and engine ignition.

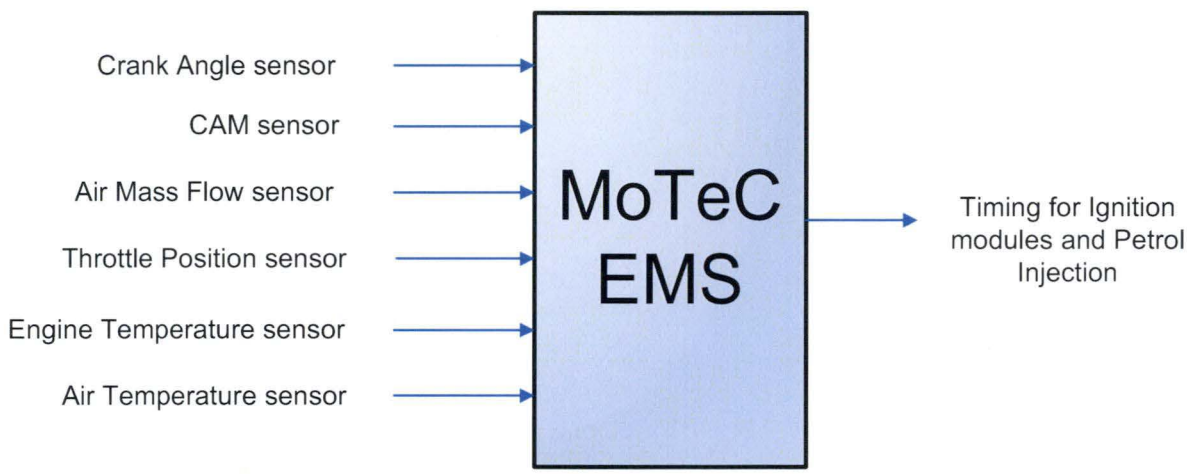


Figure 4.4: *Inputs and outputs of the MoTeC M4*

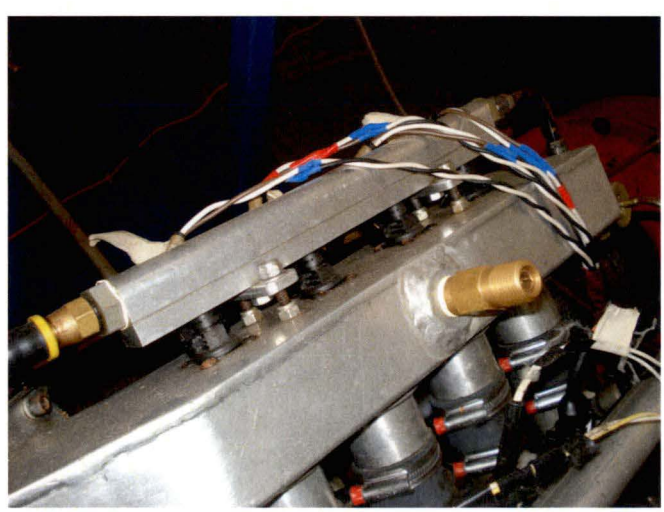


Figure 4.5: *Petrol fuel injectors*

Ignition output signals from the MoTeC Engine Management System are used to do the timing for two ignition modules in Figure 4.6. Those ignition modules are wired to cope with the setting of sequential sparking inside the MoTeC and they drive the ignition coils shown in Figure 4.7 that deliver high voltage to produce sparks.

Figure 4.6: *Ignition modules*

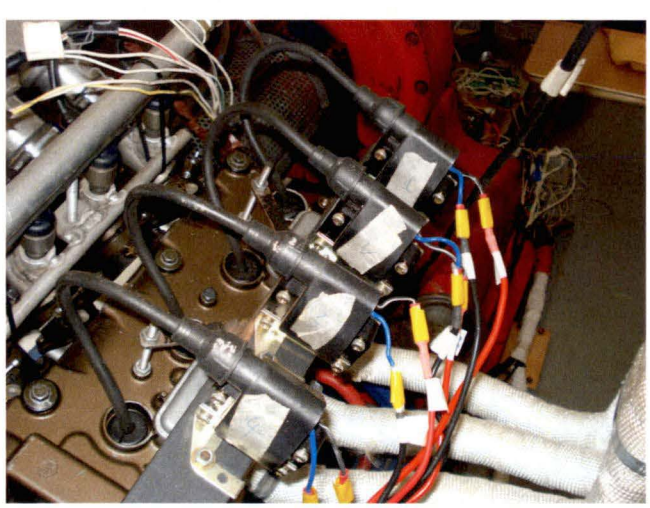


Figure 4.7: *Ignition coils*

Besides the MoTeC control unit selected for Petrol control aspect, a custom-built add-on Hydrogen injection system has been developed. This add-on system does the control of 4 Hydrogen injectors shown in Figure 4.9 and at the same time predicts the emissions online using incorporated Neural Networks. Figure 4.8 shows the components that have made up the whole control system. Design of the add-on system will be addressed in detail in the next chapter.

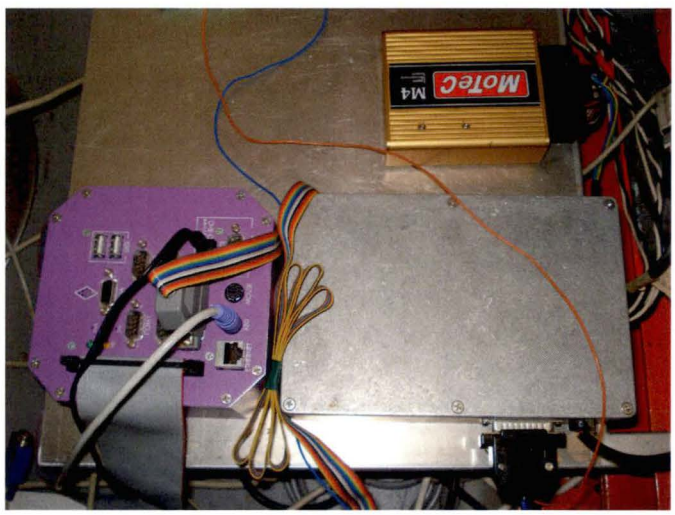


Figure 4.8: *Test rig control system*

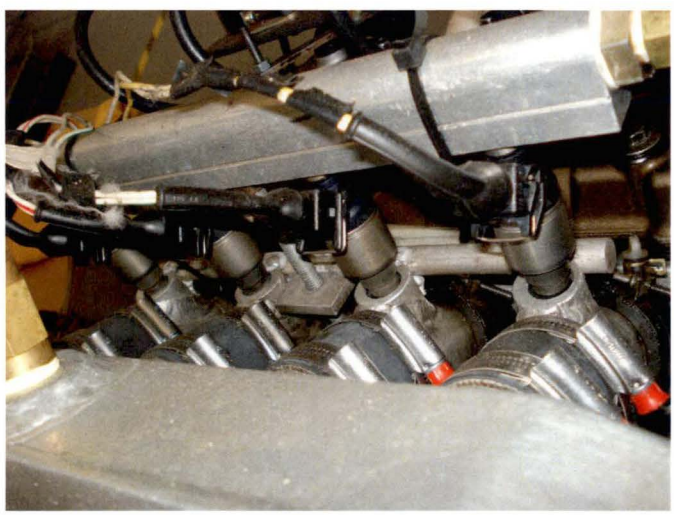


Figure 4.9: *Hydrogen fuel injectors*

4.4. ENGINE SENSORS

Test rig design also involves selection of relevant instrumentation and sensors to fit the requirements. Sensors used in this design are:

- Synchronisation sensors (including 2 CAM sensors and one crank angle sensor)
- Air Mass Flow sensor
- Throttle Position sensor
- Temperature sensors (including the Engine Temperature sensor and the Air Temperature sensor)

4.4.1. Synchronisation sensors

There are three sensors installed on the engine to provide timing synchronisation. Two Hall-effect sensors (Allegro A1382) are installed on two Cam gears as shown in Figure 4.10 and one crank angle sensor of magnetic pick-up type is mounted near the crank gear as depicted in Figure 4.11.

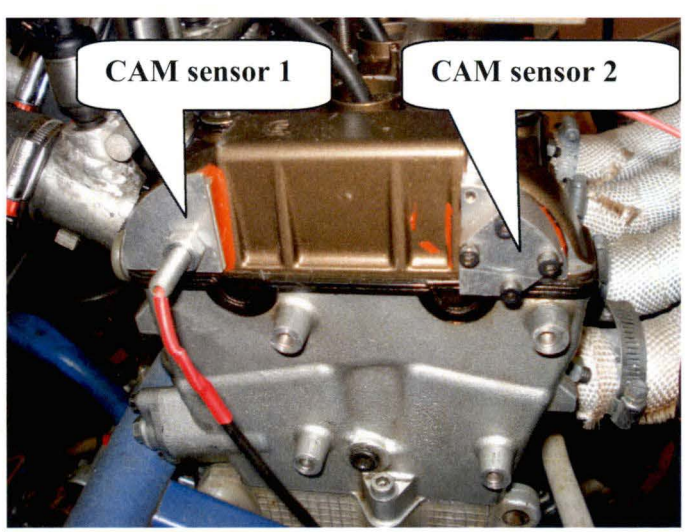


Figure 4.10: *CAM sensors*

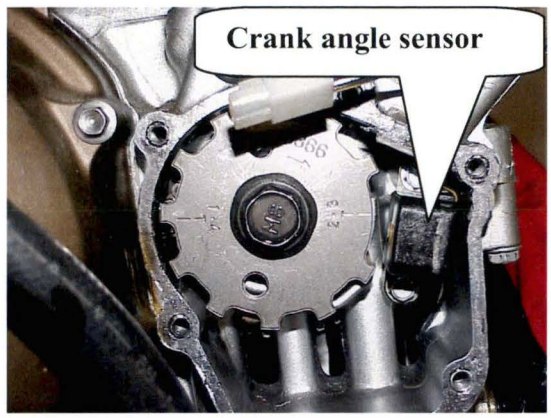


Figure 4.11: *Crank angle sensor*

CAM sensor number 2, which is referred to as the Sync signal in the sensor alignment diagram (Figure 4.12), together with the crank angle sensor (Ref signal in the diagram) are used for timing synchronisation of the MoTeC system. Since the CAM sensor number 2 is required to be mechanically positioned to detect the combustion stroke of cylinder number 1 and the rising edge of its signal must not align with either the Top Dead Center or Bottom Dead Center position of the cylinder as shown in the diagram, the CAM sensor number 1 has been utilised to provide correct timing for the add-on system.

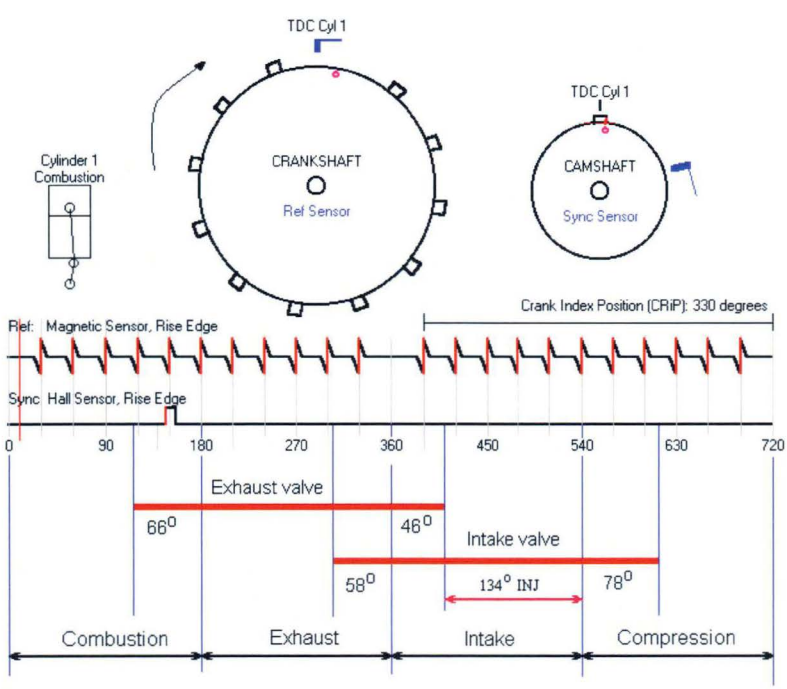


Figure 4.12: *MoTeC sensor alignment diagram*

4.4.2. Air Mass Flow sensor

An analog Bosch hot-film Air Mass Flow meter, type HFM 5 (as shown Figure 4.13), was selected to measure the mass airflow rate at the throttle. This sensor has return flow detection, which eliminates errors associated with airflow reversible caused by the reciprocating motion of the piston.

The sensor incorporates a heated element and as air is drawn into the engine the heated element dissipates its heat to the incoming air. As the airflow increases the heated element dissipates a larger amount of heated to the inducted air. The resulting temperature differential is used as a basis to determine the air mass flowing past the sensor and hence entering the engine.

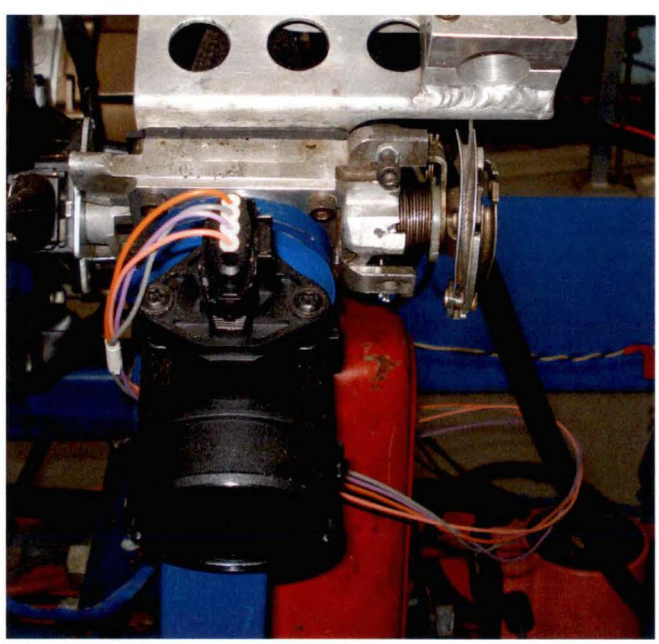


Figure 4.13: *Bosch Air Mass Flow sensor*

4.4.3. Throttle Position sensor

An analog Bosch throttle–valve angular position was chosen for measurement of the throttle position. This is a potentiometric sensor, which has a linear characteristic curve. The rotor of

the sensor is attached to the throttle valve shaft and is mounted on the throttle valve housing. As the throttle valve rotates it generates a voltage proportional to the throttle valve's angle of rotation. The installation is as shown in Figure 4.14.

Figure 4.14: *Bosch Throttle Position sensor*

This Throttle Position sensor must be calibrated using the MoTeC ECU software. The procedure involves taking a voltage reading of the sensor when it is at 0 degree (no depression on the throttle pedal) and then repeating the process when the butterfly is at wide-open position (complete depression of the throttle pedal).

4.4.4. Temperature sensors

A Delco Engine Temperature sensor is mounted on the coolant water hose (Figure 4.15) to sense the water temperature. Another Delco Air Temperature sensor is installed on the intake manifold (Figure 4.16) and near the inlet. Engine temperature and air temperature values are not used in control of either Petrol or Hydrogen injection (even they can be used for fuel trimming with respect to changes of the ambient temperature and engine operating temperature). They are used as inputs into the Neural Network models predicting engine emissions that will be discussed in the following chapters.

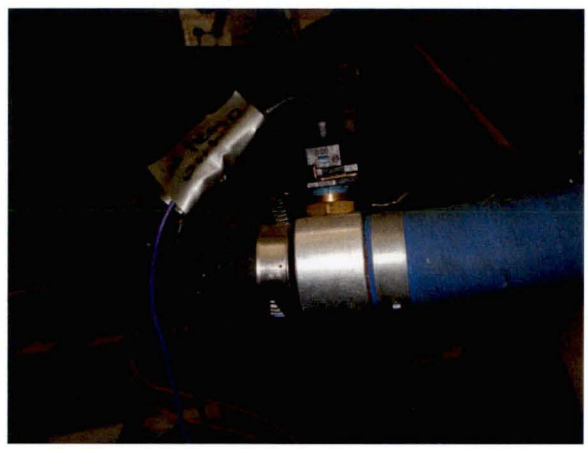


Figure 4.15: *Engine temperature sensor*

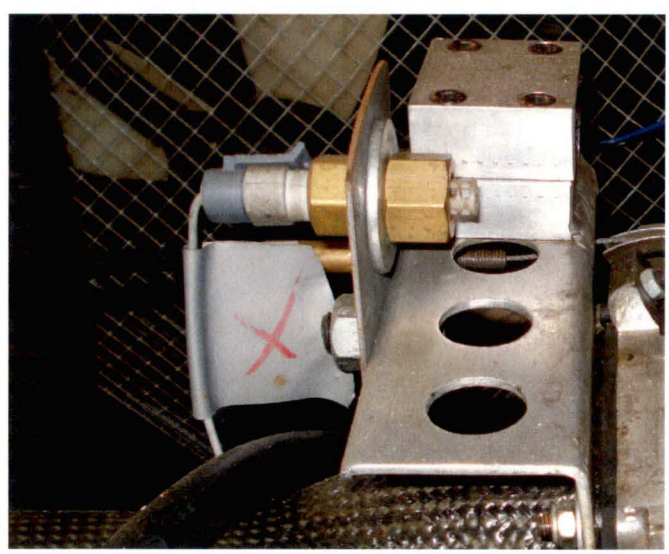


Figure 4.16: *Air Temperature sensor*

4.5. ENGINE CALIBRATION PROCEDURE

In addition to the sensors described in the previous section, this test rig is also equipped with a lambda sensor that senses the air/fuel ratio when running the engine on Petrol only. This sensor is required in the initial calibration procedure.

To enable the engine to run on mixing mode, it is required to undergo the calibration procedure. The procedure includes two phases:

- Phase 1:
  Initial tune-up of the engine to run well on Petrol with optimum fuel consumption

- Phase 2:
  Calibration of the test engine to run on mixed fuel of Petrol and Hydrogen with reduced emissions and steady operation.

4.5.1. Phase 1 – Initial calibration

Phase 1 is actually the process of finding the optimum air/fuel ratio and spark timing for engine operation on Petrol. Calibration is done with the MoTeC proprietary software. As an example, Figure 4.17 shows the calibration page that enables user to adjust the amount of Petrol to be injected on the fly.

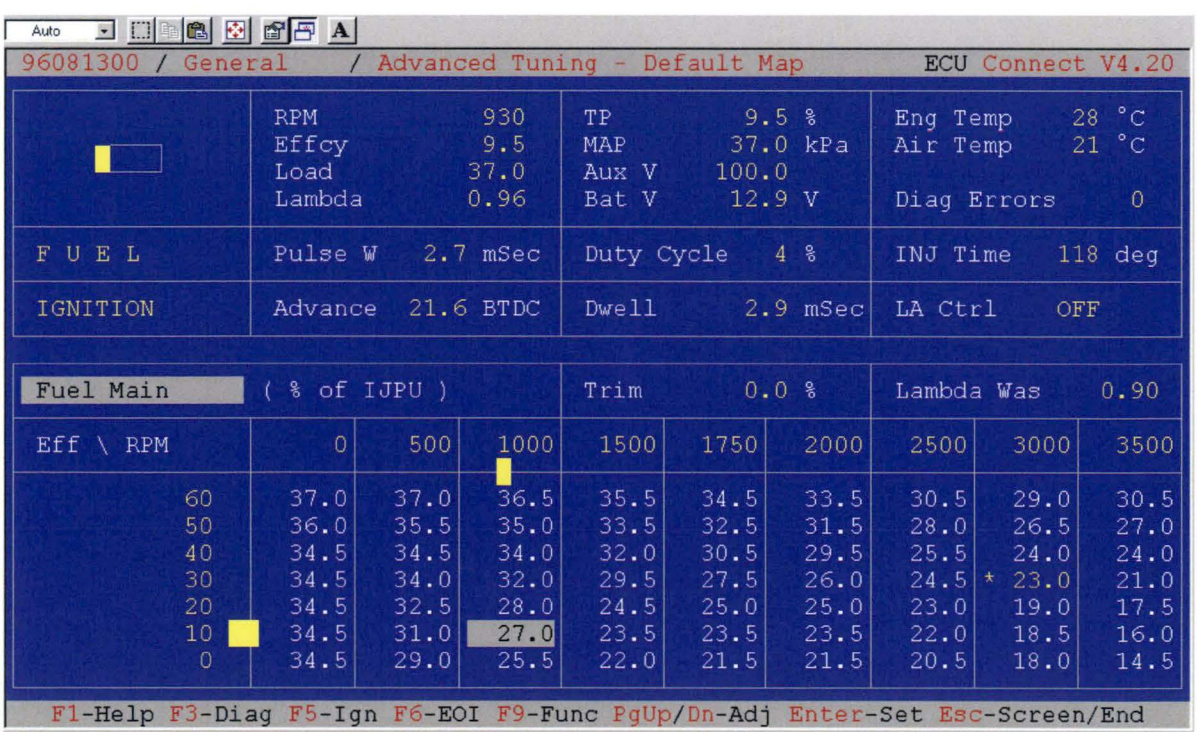


Figure 4.17: *MoTeC software interface – Injection tuning page*

The tune-up process in this phase is summarised as below:

- The ignition is set to roughly 20 degrees advance at the first two speed sites and at the first load site.

- Petrol is added at these sites by increasing the fuel duration in the fuel map as the engine is cranked until the engine fires. If the initial engine temperature is low, a degree of correction is applied to the map to enable the engine to start.

- Once started, the engine is allowed to warm up using only the first load and speed positions. By the time the engine is hot, the Air/Fuel ratio at the initial load/speed site will be trimmed to roughly the correct levels. This fuel setting can then be used as a basis for all the speed sites at the respective engine load.

- Air/Fuel ratio and ignition advance is adjusted until the engine idles at the desired engine speed (1200RPM). The timing at the speed site just above idle is set to a very low value to stop the engine from racing when at idle. Therefore, if the speed increases the timing drops back causing the speed to drop as well. Akin to that, at the speed site below idle, the timing is set to a high value to kick up the engine if the idle speed drops.

- Once the idle settings are complete, the engine is tuned for performance at other load/speed sites. The dynamometer is set at a particular RPM by adjusting the amount of water that flows into it; running the engine in a high gear and applying the throttle to hold the engine against the dynamometer until a certain target RPM is reached. At this operating point the timing for injection is adjusted until the air/fuel ratio is stoichiometric.

- If during adjustment at a particular site knocking is heard or the torque falls, ignition timing needs to be retarded.

- Once a particular engine speed and load site has been mapped, the fuelling and ignition values can be extrapolated to all successive speed sites for the particular engine load as a starting point. The load is then increased by increasing the amount of water going into the dynamometer and at the same time increasing the degree of throttle-valve opening to move to the next load point with the same speed as the previous one. Timing for the new operating point is adjusted so that the Air/Fuel is also stoichiometric and knocking is avoided.

- The process is repeated for each speed and load site that can be reached. Once the overall mapping is done the fuelling map must be smoothed out to eliminate spikes so that the engine can run smoothly.

4.5.2. Phase 2 – Mixing mode calibration

In phase 2 of the calibration process, the newly designed add-on system is run together with the MoTeC. Same actions as above are required to bring the engine to various speed and load sites. At each operating condition, using the program developed for the add-on system (Figure 4.18), the start and the duration of Hydrogen injection is specified as shown in Figure 4.19 and the amount of Petrol is at the same time reduced on the MoTeC calibration software. Tuning is done with the target of reducing the emissions and smooth operation of the test engine.

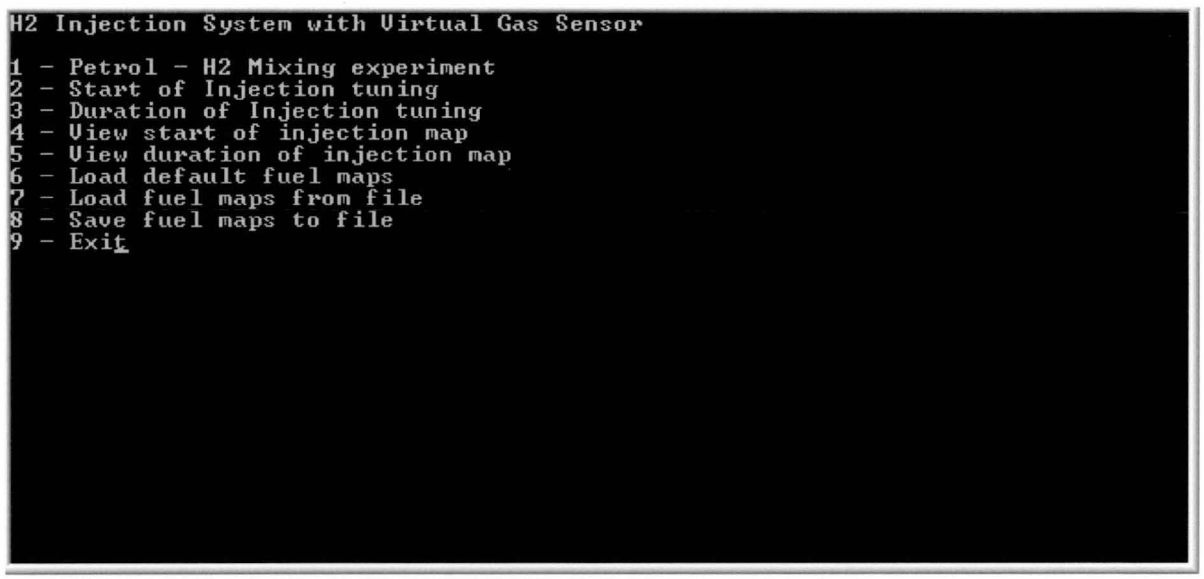


Figure 4.18: *Add-on system software*

Figure 4.19: *Add-on system injection tuning page*

Once the overall fuel maps for mixing are all established they are saved onto the corresponding control units. A copy of Petrol fuel maps are also made and stored in the add-on system. The reason for that is explained in Section 5.7.5 in the next chapter.

4.6. CONCLUDING REMARKS

This chapter has described the set-up of the experimental test engine. Basically, mechanical modification of the previously designed Petrol intake manifold has been implemented to facilitate dual fuel operation. The required sensors and instrumentation have also been installed. Those sensors are: Synchronisation sensors (including 2 CAM sensors and one crank angle sensor), Air Mass Flow, Throttle Position and Temperature sensors (including the Engine Temperature sensor and the Air Temperature sensor).

This chapter has also summarised the calibration procedure required to tune up the engine from the beginning to the stage of operating on mixing mode. The engine is initially fine tuned for optimum air/fuel ratio when running purely on Petrol. It is then calibrated together with the newly installed add-on system to operate on mixing mode with Petrol plus Hydrogen.

CHAPTER 5

EMBEDDED ADD-ON FUEL INJECTION SYSTEM WITH VIRTUAL EMISSION SENSOR

5.1. INTRODUCTION

In the system described in the previous chapter, design of the add-on Hydrogen fuel injection system with integrated emission prediction capability plays a key role in this research. This system is required for data collection, control of Hydrogen injection and quantitative prediction of the exhaust gas.

The add-on system is comprised of two separate modules: a PC104 computer and a Motorola 68HC12 microcontroller. Those boards are required to be programmed with appropriate control algorithm. Programming and design of the whole system will be discussed in detail in this chapter. In addition to that, relevant concept of injection control will also be provided. The control programs have been developed in C and source codes are provided in the Appendix.

5.2. CONTROL OF FUEL INJECTORS

Based on the types of their driver circuitry, injectors are basically classified into 2 types:

- "Saturation" fuel injectors.
- "Peak and hold" fuel injectors

5.2.1. "Saturation" fuel injectors

With a "Saturation" fuel injector, a 12 V saturated circuit driver (Figure 5.1) is needed to control it. Injectors of this type have high-resistance, typically 12-16 ohms.

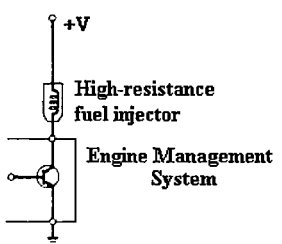


Figure 5.1: *High-resistance fuel injector set-up*

The current flowing through the injector is low, keeping the components nice and cool for long life since the design reduce heat build up and heat dissipation in the control circuit. Figure 5.2 shows the profile of electrical current flowing through injectors of this category.

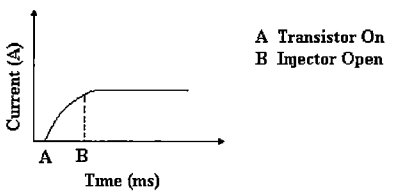


Figure 5.2: *High-resistance fuel injector current*

However, injectors of this type have slower response time and closing time than the "Peak and hold" injectors.

5.2.2. "Peak and hold" fuel injectors

A "Peak and hold" fuel injector has very low resistance, normally 2-5 ohms. Those of this type are also referred to as high-flow injectors and primarily used in after-market high performance systems. The "Peak" is the current required to pass through the solenoid in order to lift the pintle inside the injector (to open the fuel injector) and the "Hold" is the current needed to hold the solenoid so that the fuel injector is kept open in a desired period of time. The current to open the fuel injector is always higher than the current to hold it open.

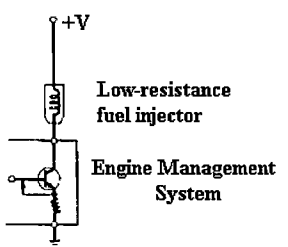


Figure 5.3: *Low-resistance fuel injector set-up*

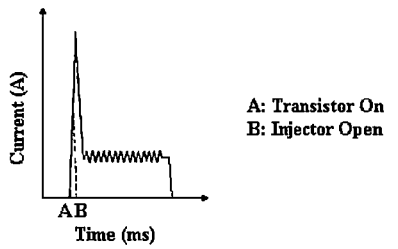


Figure 5.4: *Low-resistance fuel injector current*

Battery voltage is applied to the injector until a predetermined current level is reached. The current is then reduced and held during the duration of the pulse width. Low-resistance injector control setup is depicted in Figure 5.3 where the feedback in the transistor denotes the sensing mechanism and is essential for all fuel injectors of this type. Figure 5.4 shows the profile of injector current of this type.

The Hydrogen fuel injectors chosen for this project are provided by QUANTUM. They are low-resistance device requiring a peak and hold drive circuit.

5.3. ELECTRONIC TIMING FOR INJECTION CONTROL

For engine control timing, it is necessary to know the position of Top Dead Centre of one among 4 engine cylinders for reference. Acknowledgement of a cylinder's position could be achieved using a Hall-effect sensor.

The sensor is working based on the principle: "When a current-carrying conductor is placed into a magnetic field, a voltage will be generated perpendicular to both the current and the field".

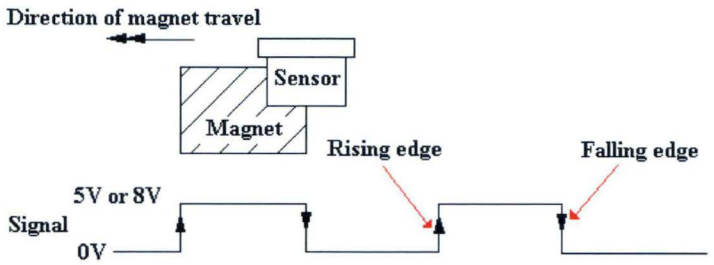


Figure 5.5: *Hall-effect sensor*

As shown in Figure 5.5, when the first side of the magnet passes the surface center of the sensor, the output signal would jump to HIGH level (5V or 8V). When another side of the magnet passes the sensor center, the signal would jump to LOW level (0V). As a consequence, rising edge and falling edge of the signal correspond to the sides of the traveling magnet.

The Hall-effect sensor is mounted on the top engine cover to sense the magnet surface installed in the cam-gear. It is adjusted so that the instance when cylinder number 1 (the reference cylinder) reaches its Top Dead Center position, starting the injection stroke, coincides with the rising edge of signal from the sensor. The signal from the Hall-effect sensor is also denoted as the CAM signal.

Due to the fact that a CAM pulse occurs every engine cycle (2 crank revolutions), the signal can be used as reference for timing engine injection. In one engine cycle, the lag time of the injection of one cylinder, e.g. cylinder number 1, with respect to the rising/falling edge of the CAM pulse, as well as the pulse-width are calculated once. Consequently, waveforms of injection/ignition signals for other cylinders have the same pulse width as the first one. All the injection/ignition signals are 180^0 different in phase.

The test engine has the firing order 1-2-4-3 (i.e. the firing order of cylinders of the engine), therefore its timing pattern for injection is as shown in Figure 5.6.

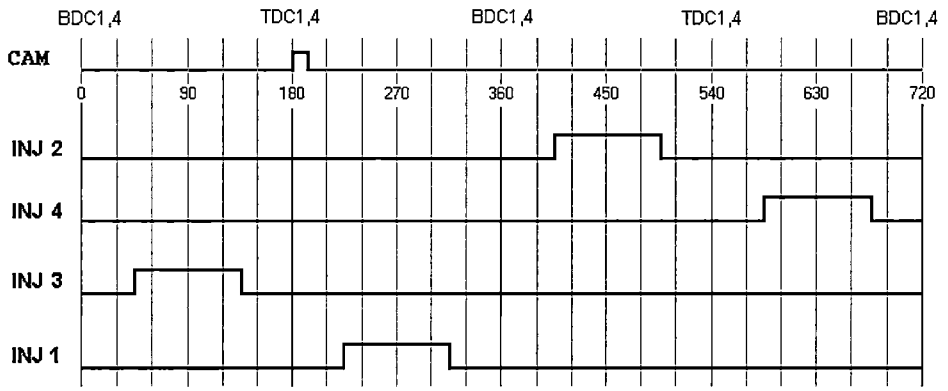


Figure 5.6: *Injection timing pattern*

5.4. DESIGN LAYOUT

The embedded add-on Hydrogen fuel injection system comprises a PC104 board and a Motorola 68HC12 micro-controller board as shown in Figure 5.7. The overall system is synchronised by the signal from the CAM sensor mounted on the cylinder head's cover. Embedded C code for the 68HC12 micro-controller has been programmed and compiled in ImageCraft cross compiler (for Motorola boards) while code for the PC104 board has been programmed in Turbo C 3.0 compiler.

The PC104 board was chosen because of its embedded data acquisition capacity and its processing power integrated in a compact module. It acts as a "brain" of the control system. The board samples data from the air temperature sensor, the engine temperature sensor, the throttle position sensor, the air-mass flow sensor and also calculates the engine speed from the CAM signal. Based on those measures, depending on the control C code pre-programmed in the flash memory (which can potentially be neural network code), certain algorithm is performed and the two parameters for injection control of the Hydrogen flow are extracted: the start of injection after Top Dead Centre and the duration of injection. In addition to generating those two control values, this board can also be programmed to perform prediction task and acts as a virtual sensor with incorporated AI tools.

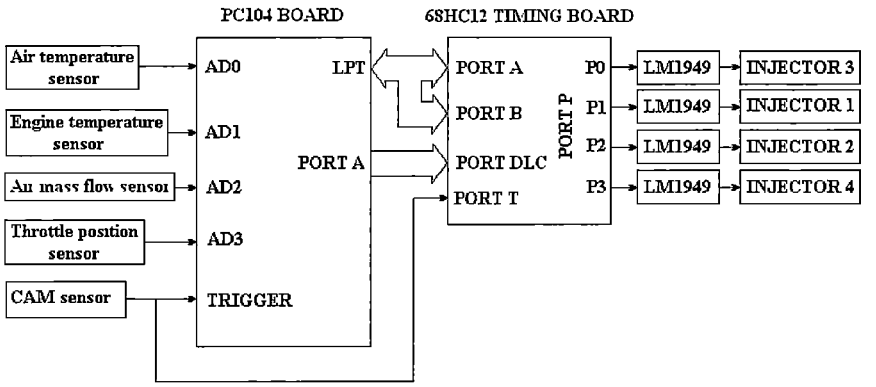


Figure 5.7: *Injection controller design layout*

The start of injection and the duration of injection of Hydrogen in terms of crank angle are passed through to the HC12 timing board via the LTP port of the PC104 board. Since there are only two packets of data to be transmitted, this method was chosen for simplicity. Each of the parameters in the form of 8-bit data packet is assigned a 2-bit ID and sent to the timing board in a hand-shaking manner (i.e. using two signals denoted as STROBE and ACKNOWLEDGE to inform the two boards when to read information and when the reading process is finished). Port A of the PC104 board is used to inform the timing board of the state of the engine, in another word is whether the timing board should operate the fuel injectors or not.

At the HC12 timing board, the Start of Injection (SOI) and Duration of Injection (DOI) data packets are received via the PORT A and PORT B of the board in terms of crank degree. The internal program then interprets the piece of data to digital logic control signals at PORT P to control the fuel injectors. Those logic signals are sent to driver board designed based on the LM1949 chip. LM1949 injector driver chip has in-built current sensing mechanism and is responsible for driving the low-impedance Hydrogen fuel injector with proper peak and hold currents. Figure 5.8 shows the design of an electronic driver circuit for one fuel injector. The circuitry has been designed in Protel and is based on the circuit provided in the manual sheet of LM1949 chip.

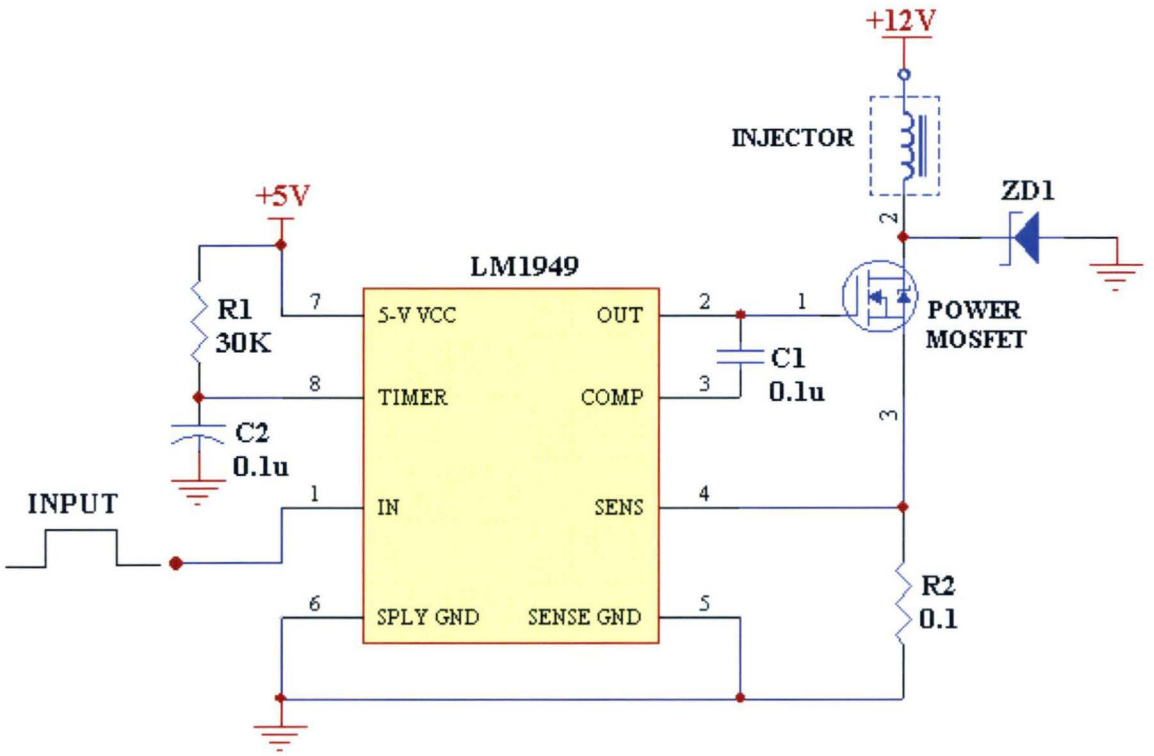


Figure 5.8: *Injector driver circuit*

5.5. TIMER SYSTEM OF THE MOTOROLA 68HC12 MICROCONTROLLER

The M68HC12 timer system is based on the 16-bit free running counter called TCNT. The TCNT register counts from 0 (0x0000) to 65535 (0xFFFF) with a frequency previously set by the three bits PR0, PR1, PR2 of the TMSK2 register. This TCNT register can be read any time to provide time information. When the register exceeds a count of 65535 pulses (0xFFFF), the counter overflows and is reset to 0 (0x0000). At this instance, a timer overflow flag (TOF) is set. Through this operation, the controller is able to maintain synchronisation and control of tasks over duration longer than the equivalence of 65535 pulses. After an overflow, i.e. the TOF flag is set to 1, in order to detect future occurrence of an overflow, the TOF flag must be cleared. The counter is reset when the micro-controller is reset and runs continuously in general. The Input Capture (IC) and Output Compare (OC) functions are important features of the timer system for the 68HC12 micro-controller. Port T of the 68HC12 is associated with Input Capture/Output Compare.

When a pin on port T is programmed as an Input Capture pin, an interrupt is generated on a rising or falling edge (or both of them) of the input signal to the port. The timer module then captures the current value of the timing register (TCNT). The captured value of the TCNT register is stored in the Timer Input Capture/Output Compare register TCn (n=0,1,...,7) which is a register associated with the corresponding nth pin of Port T. An example of using the Input Capture function is to measure the pulse-width of an incoming digital signal, or to accurately measure the time of an event, such as in time-of-flight measurements of an ultrasonic range-finder.

When a pin on Port T is programmed as an Output Compare pin, a value is loaded into the corresponding TCn register for comparison with TCNT. When a successful comparison is made (between TCn and TCNT), an interrupt is generated. In another word, the interrupt occurs when the value in the TCn register associated with the nth pin of Port T equals the value of the TCNT register. The nth pin in Port T, when an interrupt happens, can be disconnected, toggled, cleared or set. Examples of the use of the Output Compare function include generating a wave form with desired frequency and duty cycle or making precise timing applications.

Whenever an interrupt due to an edge occurrence in Input Capture or a successful comparison in Output Compare is generated, the corresponding flag CnF (n=0...7) in the Timer Interrupt Flag 1 (TFLG1) register is set and the flag must be cleared in the service routine for the next interrupt to happen.

5.6. TIMING CALCULATION FOR THE MOTOROLA 68HC12 BOARD

The injection timing board has been designed to work in the range of engine speed greater than 600 rpm. It implies that the add-on Hydrogen injection system is only triggered after the engine has already been cranked up on Petrol. Specifications for the test engine stipulate that the maximum engine speed can be as high as 13000 rpm. At minimum speed (600 rpm), the CAM signal has a period of 200ms (5Hz) over 702 crank angles. At 6000 rpm, the CAM signal has a period of 20 ms (50Hz) as in Figure 5.9.

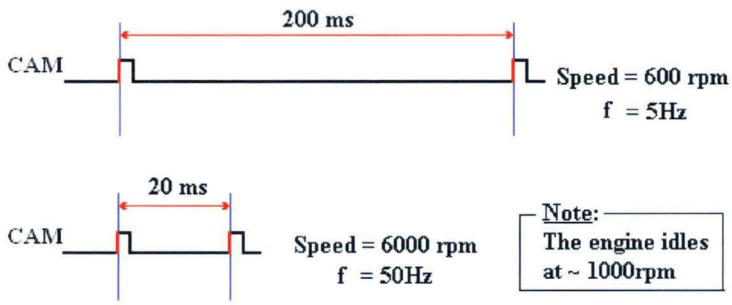


Figure 5.9: *CAM signal at 600 rpm and 6000 rpm over 720 crank angles*

Engine control signals are synchronized and timed using an internal clock in the Standard Timer Module of the 68HC12 board. The 16-bit clock register counts freely from 0 (0x0000) to 65535 (0xFFFF) and rolls back to 0 in a preset period of time T.

In order to have (at most) only one rollback of the counter during 720^0 (two revolutions), period T has to be greater than the maximum period of the CAM signal (200 ms).

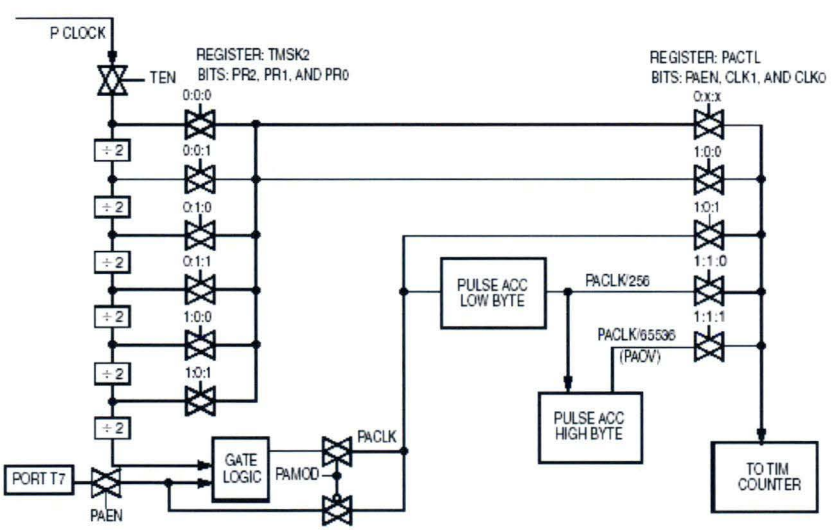


Figure 5.10: *Clock chain of the Timer Module of the 68hc12 board* [4].

Figure 5.10 shows the clock chain of the Timer Module with the main clock is PCLOCK. Setting bits PR2, PR1, PR0 of the TMSK2 register would successively divide the main clock

with the appropriate factor to give the actual clock frequency that drives the free running counter register TCNT.

With the PCLOCK frequency of 8 MHz and the setting of bits PR2=1,PR1=0 and PR0=1, the PCLOCK frequency would be divided by a factor of 25. As a consequence, the 16-bit freely-running counter (TCNT register) would have a frequency of 8/25 = 0.25 MHz. Therefore, the counter register (TCNT) counts from 0 to 65535 in (1/0.25MHz)*65536 = 262144 μs = 262.144 ms. This counting period is greater than the maximum period of the CAM signal when engine speed is at the lowest 600 rpm, ensuring the full coverage of clock counts over one engine cycle (720^0) for control signal timing with at most one counter overflow.

+ At engine speed = 600 rpm (f_{CAM} = 5Hz,T_{CAM} = 200ms):

Number of clock ticks covering 200 ms:

$$N1 = \frac{200ms * 65536ticks}{262.144ms} = 50000ticks$$

Resolution:

$$R1 = \frac{50000ticks}{720^0} = 69.44ticks/\deg ree$$

+ At engine speed = 6000 rpm:

Number of clock ticks covering 200 ms

$$N2 = \frac{20ms * 65536ticks}{262.144ms} = 5000ticks$$

Resolution:

$$R2 = \frac{5000ticks}{720^0} = 6.94ticks/\deg ree$$

Timing of the injection is determined at every rising edge of the CAM signal. The parameters of engine injection are "Start of Injection" and "Duration of Injection". Significance problem arises when dealing with resolution calculation (number of 68HC12's timer clock ticks corresponding to 1^0 crank degree at different engine RPM). With a theoretical resolution of 6.94 ticks/degree at 6000 RPM, the resolution calculated by a micro-controller would be interpreted as 6 since the decimal part is truncated in an unsigned short number division. An unsigned short number is a 2-byte number and ranges from 0 to 65535. As a consequence, the truncated decimal part causes huge errors in the translation from crank degrees to number of clock ticks while accuracy is a priority in engine control timing.

Another possible solution to the problem is using float numbers in the calculation. Float numbers are 4-bytes numbers, in the range from +/-1.175e-38 to 3.40e+38. At a particular speed, instead of taking into consideration of the number of clock ticks per 1^0 of crank angle, calculating the resolution in terms of the number of clock ticks per 100^0 of the engine crank would preserve the accuracy. Correspondingly, a value of engine crank angle when to be converted to the number of clock ticks is given by

$$Angle_{tick} = (Angle_{degree} * Resolution\ ) \ / \ 100$$

The 68HC12 micro-controller handles the new computational method well, hence timing is updated at the start of every engine cycle and precision is greatly improved.

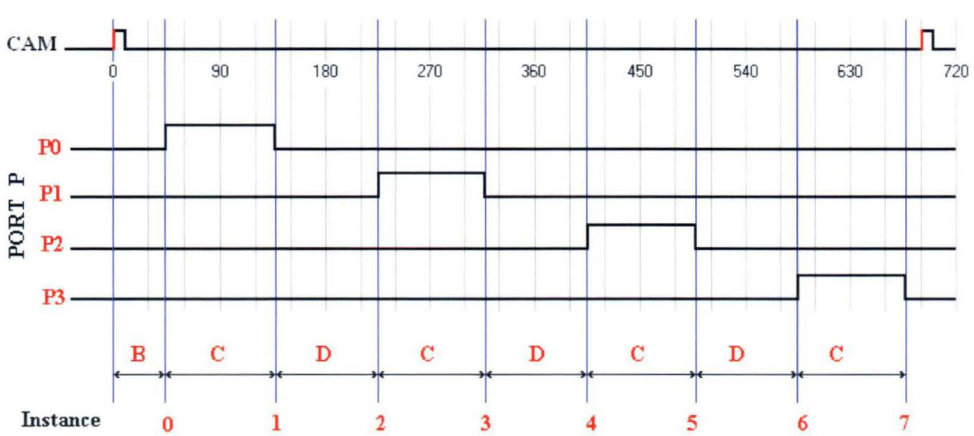


Figure 5.11: *Injection control logic signals*

Controlling 4 fuel injectors requires 4 digital I/O lines. Figure 5.11 shows the signal waveforms in one engine cycle of 720^0 (2 crank revolutions). The relative (angular) position of the Hall effect sensor (around the camshaft) with respect to the position of the pistons inside the cylinders is configurable. This can be fixed during the mounting of the Hall effect sensor (and its pick-up magnet) on the engine. For the test engine, the position of the Hall effect sensor signal (rising edge) corresponds TDC of the reference cylinder.

Injection of the cylinder number 1 starts at B degree after it moves away from Top Dead Centre and controlled by pin P0. Duration C of the injection is the same for all the 4 injectors. The start of injection for the cylinder number 2, controlled by the I/O line P1, is D degree (180^0 - C) after the end of injection controlled by P0. Similarly, the start of injection controlled by P2, P3 is D degree after the end of injection at P1, P2, respectively. Overall injection timing can be divided into 8 instances and used as reference when programming the micro-controller to generate desired control signals.

The general purpose I/O Port P of the 68hc12 board is used to generate injection control signals. The rising edge of the CAM pulse is captured by the Input Capture on pin 4 of the timer Port T. Pin 0 of Port T is configured to be Output Compare. Hence, its associated timer register TC0 is used to monitor instances shown in Figure 5.11 to set the logic level of pin 0,1,2,3 of Port P properly to generate desired signal waveforms. Whenever the rising edge of the CAM pulse is detected, an interrupt is generated and engine rpm, injection start angle, injection duration and lag angle in terms of clock ticks would be calculated. Then, the value of the start angle B in clock ticks, added with the current value of TCNT, is loaded into the Output Compare TC0 register. Therefore, an interrupt is generated when TCNT=TC0 and is identified as instance 0 (corresponding to the start of injection for the reference cylinder) and the logic level of pin P0 is 1. At this instance, the duration angle C in clock ticks, added with the current TCNT, is loaded into TC0 to have an interrupt generated at instance 1, at which the level of pin 0 of Port P is set to 0 indicating the end of injection for the first injector. Continuous monitoring the instance variable ensures accurate timing for injection of all the four cylinders of an engine.

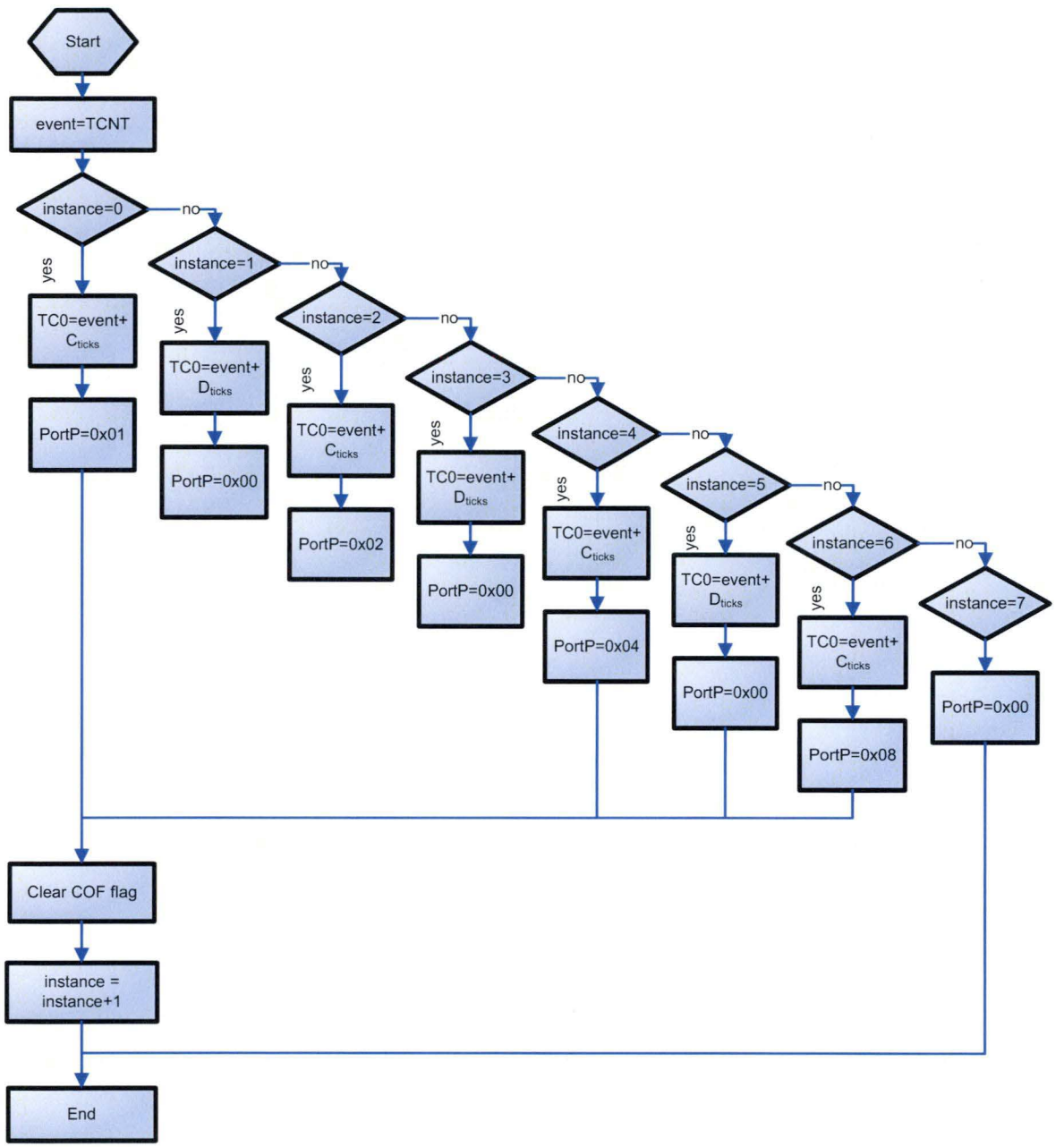


Figure 5.12: *Interrupt routine attributed to Output Compare TC0*

Figure 5.12 represents the interrupt routine attributed to the Output Compare TC0 and fires whenever a successful comparison in the register is made. It monitors the instance mentioned above to decide appropriate timing output at port P.

5.7. CONTROL LOGIC OF THE PC104 BOARD

The add-on Hydrogen injection system employs conventional control methodology to determine the start of injection and the amount of Hydrogen to be injected at each operating point of the test engine. Conventional control method uses a number of look-up tables to store control data. A look-up table is a 2-dimensional table as shown in Figure 5.13 with one axis being the speed of the engine (RPM). Another axis represents the load of the engine, which can be calculated from the Throttle Position (in terms of percentage, 100% when it is fully opened).

		Speed (RPM)			
		1000	**1500**	**2000**	**2500**
Load (%)	**60**	30	29	20	20
	50	29	29	25	19
	40	28	26	22	18
	30	26	25	23	18
	20	26	24	24	19
	10	20	23	25	27

Figure 5.13: *Look-up table*

The PC104 board stores 4 look-up tables as 2-dimensional arrays in the flash memory, two tables contain the start and the duration of injection of Hydrogen and two similar tables contain the start and duration of injection of Petrol when engine running on mixture of Petrol and Hydrogen. The last two tables are duplicated from the MoTeC control system and are used for emission prediction (Petrol injection is controlled by the MoTeC unit).

The flowchart in Figure 5.14 below shows the algorithm of the control program integrated inside the PC104 board.

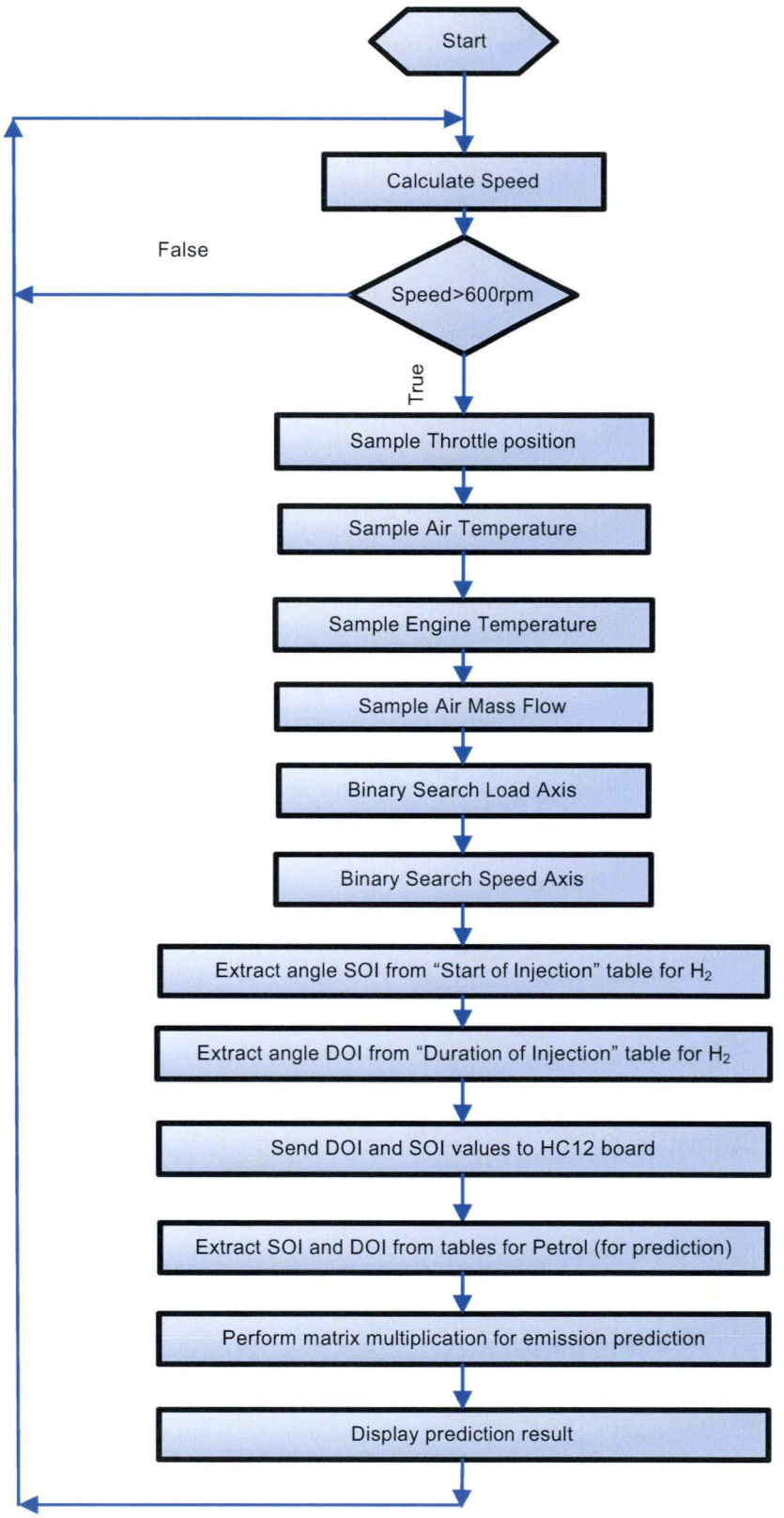


Figure 5.14: *Add-on system algorithm*

5.7.1. Speed calculation

Apparently, speed is not simply a number that can be read from a sensor but needs to be calculated. The PC104 board has used a 16-bit 100kHz countdown counter to do this job. An interrupt routine (i.e. running separately from the main program) has been set up to be triggered by a rising-edge of the CAM signal described in section 5.3. The routine when fired would extract the Most Significant Byte and the Last Significant Byte of the 16-bit counter, and then it sets the counter to FFFF (65535). Finally, the routine enables the counter to continue its countdown operation. The algorithm is shown in Figure 5.15 below.

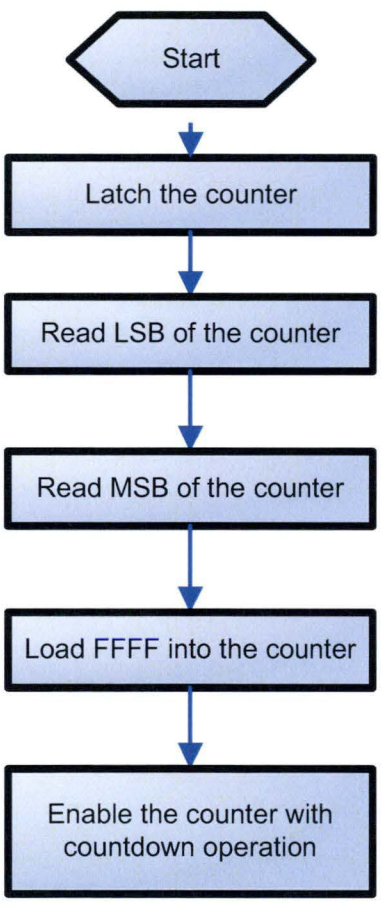


Figure 5.15: *CAM rising-edge-triggered routine*

Whenever the main program requires the speed value, it picks up the Most Significant Byte (MSB) and the Least Significant Byte (LSB) extracted above. The number of clock ticks corresponding to the engine speed is given by:

$$N_{ticks} = 65535 - (256*MSB - LSB) \qquad (5.1)$$

Given the fact that every Camshaft rotation corresponds to 2 engine revolutions and the clock speed of the counter is 100kHz (or 100,000Hz), the engine speed is calculated by:

$$N_{RPM} = \frac{2*60*100000}{N_{ticks}} \qquad (5.2)$$

In the mixing mode, injection of Hydrogen is only enabled when the engine has already cranked on Petrol and engine speed is greater than 600RPM. The limit 600RPM was imposed due to limitation the timing board discussed in Section 5.6.

5.7.2. Analog data sampling

There are four engine parameters that need to be sampled during engine operation. They are:

- Air Mass Flow
- Air Temperature
- Engine Temperature
- Throttle Position

The PC104 board has a built-in 16-bit Analog-to-Digital Conversion (ADC) unit that allows 16 single-ended or 8 differential analog inputs from sensors. In single-ended mode, all the sensors have common ground while the differential mode requires separate earth from each senor. The single-ended configuration was not selected since it is prone to noise.

Due to hardware configuration, values sampled from sensors are in the range –32767 to +32767. Therefore the real values of each signal in Voltage is given by:

$$Value_{volt} = \frac{(Value_{sample} + 32768) * Range}{65536} \qquad (5.3)$$

where

Range = the scanning range of ADC corresponding to full scale of 16-bit and is equal to 5V.

The Throttle Position value in Volt (V) is required to be converted to percentage for further processing operation.

$$Value_{\%} = \frac{(Value_{volt} - MIN_{volt}) * 100}{MAX_{volt} - MIN_{volt}} \qquad (5.4)$$

where

MIN_{volt} = throttle position in Volt corresponding to fully closed position

MAX_{volt} = throttle position in Volt corresponding to fully open position

In the equation above MIN_{volt} and MAX_{volt} are measured only once during preliminary calibration.

5.7.3. SOI and DOI extraction from look-up tables

There are two key parameters required for Hydrogen injection control: the Start of Injection (SOI) and the Duration of Injection (DOI). Those parameters are stored in two different look-up tables named "Start of Injection" and "Duration of Injection". Those tables are established during initial calibration. Each table has the form as described in Figure 5.12. One axis of the table represents the engine load, which is also the Throttle Position in percentage, and another one is Engine Speed. Due to the fact that the number of operating points stored in the table is limited, the system, when reading a particular pair of data of Throttle Position and Engine

Speed while the target engine is running, needs to determine the operating point in the pre-programmed look-up tables closest to the current engine operating condition so that appropriate control parameters could be picked up for proper injection timing. This processing step uses Binary Search since the Load and Speed axes store values in monotonically increasing order and is illustrated below:

The "Start of Injection" look-up table contains the crank angle that Hydrogen injection occurs. Its Speed axis in RPM contains the following values: 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, and 5000. The Load axis contains values in the range from 0% to 100% with an increment of 10% for each value. If the engine is running at 2200RPM, the control program performs a Binary Search and determines the corresponding operating Engine Speed in the look-up table being 2000. If the actual Throttle Position is 25%, the Binary Search algorithm will return a corresponding pre-programmed Throttle Position of 20%. Based on those results, the control program then picks up the angle in the look-up table that decides when the injection of Hydrogen takes place. The angle is referred to as SOI.

The same process is implemented for the "Duration of Injection" table to pick up the crank angle that settles how long the injection of Hydrogen would last. The resulted angle is referred to as DOI.

5.7.4. Transfer of SOI and DOI to HC12 board

Once the Start of Injection (SOI) and Duration of Injection (DOI) parameters are extracted from look-up tables, they need to be transmitted to the HC12 board as 8-bit integers. As mentioned in Section 5.4, the PC104 board uses LPT port to transfer those numbers to the HC12 board. Each of the parameters in the form of 8-bit data packet is assigned a 2-bit ID. AFEED and INIT pins of the LPT port are used to signal the HC12 board which packet is being sent as in Figure 5.16.

INIT	AFEED	Data
0	1	SOI
1	0	DOI

Figure 5.16: *Data packet ID assignment*

In addition to that, the STROBE and ACKNOWLEDGE pins of the LPT port are used to indicate the transfer status. Before a packet is sent, the corresponding ID assignment is set on INIT and AFEED pins. After that, the STROBE is set to 1 to indicate the HC12 board that a data packet has been placed on the 8-bit data bus of the LTP port. When the HC12 board has successfully read the data, it pulls up the ACKNOWLEDGE pin of the LPT port to inform the PC104 that the transfer has succeeded and it is ready for the new packet. This mode of data transfer is referred to as "Hand-shaking" and is chosen for this system due to its simplicity and quickness.

5.7.5. Matrix calculation for emission prediction

Due to the fact that MoTeC Engine Control Unit (ECU) is a "black-box" system, data that it acquires and processes cannot be retrieved by a "third-party" system. In other words, using the PC104 to extract control data for Petrol injection from MoTeC is not possible. Therefore the injection look-up tables for Petrol in the MoTeC system are duplicated and stored in the PC104 board. Using those tables the control program in the PC104 can be aware of when the injection of Petrol happens and how long that injection is. That information is then fed into the calculation for emission prediction. Extraction of those parameters from look-up tables is similar to the process described in Section 5.7.3. The emission prediction capability of the system is also referred to as a virtual emission sensor.

In order to perform the prediction task, the system requires the network input vector **X** containing the following parameters:

- Engine Speed (RPM)
- Throttle Position (V)

- Air Temperature (V)
- Engine Temperature (V)
- Air Mass Flow (V)
- SOI of Hydrogen (Degree)
- DOI of Hydrogen (Degree)
- SOI of Petrol (Degree)
- DOI of Petrol (Degree)

Outputs resulted from the prediction task are represented by the output vector **Y** containing the parameters below:

- CO (%)
- HC (ppm)
- NO (ppm)

Two neural network models incorporated into the prediction scheme for online testing are Optimisation Layer by Layer and Hybrid with only one hidden layer. Both types use sigmoid activation function in the hidden layer and linear activation function in the output layer. The offline training process of those network results in 2 weight matrices for each model. One is the matrix $\mathbf{W}_{\mathbf{hidden}}$ of the connection between the input layer and the other is of the connection between the hidden layer and the output layer called $\mathbf{W}_{\mathbf{out}}$.

The procedure of calculating emissions for one particular network model is summarised as:

- Normalising inputs to the same scale from 0 to 1
- Calculating output of the hidden layer

  $$\mathbf{N} = \text{sigmoid}(\mathbf{W}_{\mathbf{hidden}} \cdot \mathbf{X})$$

  where

  $\mathbf{N}$ = vector containing outputs from hidden neurons.
- Calculating output of the output layer

  $$\mathbf{Y} = \mathbf{W}_{\mathbf{out}} \cdot \mathbf{N}$$

Values of vector **Y** are then displayed onto the screen if a monitor is connected to the PC104 board.

5.8. CONCLUDING REMARKS

This chapter has provided detail of the design and associated control methodology of an embedded add-on Hydrogen fuel injection system with incorporated neural network virtual sensor.

The system is essentially the combination of a PC104 computer board and a Motorola HC12 board. The PC104 board plays the role of the brain in the system. It samples signals from external sensors and determines the amount as well as the instance that Hydrogen injection occurs during engine fuel mixing operation. The Motorola HC12 board basically converts the decision made by the "brain" in terms of numbers into electronic timing signals to drive the driver circuitry that controls Hydrogen fuel injectors.

Besides its control capability, the PC104 board also performs the matrix calculation to predict the emissions based on the weights resulted from the training of two neural models: the Optimisation Layer by Layer and the Hybrid networks.

CHAPTER 6

PREDICTIVE MODELS FOR EMISSIONS IN PETROL-HYDROGEN DUAL FUEL ENGINE USING NEURAL NETWORK

6.1. INTRODUCTION

In this chapter, various Neural Network models employed for emission prediction task will be discussed. The common procedure of modeling will be presented and the modeling processes for CO, HC and NO will also be described. Assessment of those models will be conducted and the resulting Neural Network weight matrices will later be incorporated into the add-on control system that has already been discussed in the previous chapter.

Neural Network modeling process in this project is implemented on the *Neural Network Analysis Package* that has been developed at the School of Engineering – University of Tasmania by various researchers including the author of this thesis.

6.2. NEURAL NETWORK ANALYSIS PACKAGE

Neural Network Analysis Package 's user interface is based on Visual Basic macros running on Microsoft Excel background and the kernel that does most of the Neural Network processing tasks is programmed in Pascal. The software also automates initial preparation of raw data. Initial data manipulation includes formatting data patterns, normalising data, selecting training and testing data, and preparing data text files for network training and testing. Various neural network models shown, from software, can be selected and their network parameters can also be varied. Evaluation of results such as calculation of the RMS error, comparative graphs and summary of percentage prediction error can also be carried out with this software.

The summary below describes the processing steps when conducting analysis with this tool:

- Worksheets are opened in a format ready to be filled with data patterns from a database.

- The software normalises data patterns to the range from 0 to 1 and creates a worksheet to save them without overwriting the original data patterns.

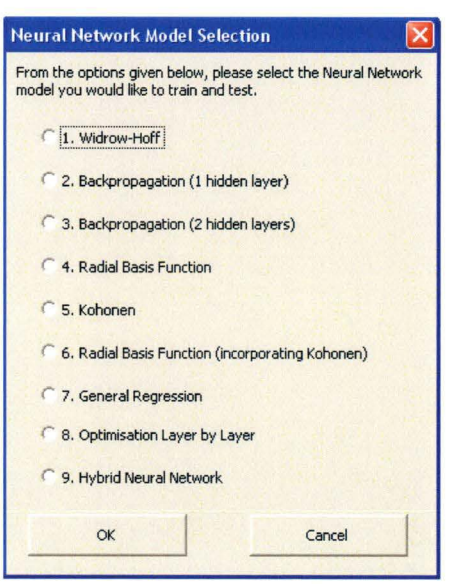
- The normalised data patterns are separated into 2 groups: training and testing data. The split can be manually or randomly selected.

- A text file of training data and one of testing data are created in a compatible format to feed to executable programs that are used for training and testing Neural Network models.

- In a given application, a neural network model is chosen from the list shown in the software snapshot below. Specific network parameters corresponding to the selected one would then have to be specified such as the number of hidden neurons, the number of iterations…

- Network training and testing are then performed. As the training and testing finish, output files are generated.

- Results can then be uploaded and Network output analysis methods are then available to users.



Network Models featured in Neural Network Analysis Package

6.3. GENERAL PROCEDURE OF NEURAL NETWORK PREDICTIVE MODELING

Exhaust gas emissions have been measured using an ADS 9000 Super Four Gas Analyser. It measures the concentrations of CO (%), HC (ppm), and NO (ppm). Each of those gases has been modeled based on the collected experimental data and each models consists one layer with a number of inputs as shown in Figure 6.1.

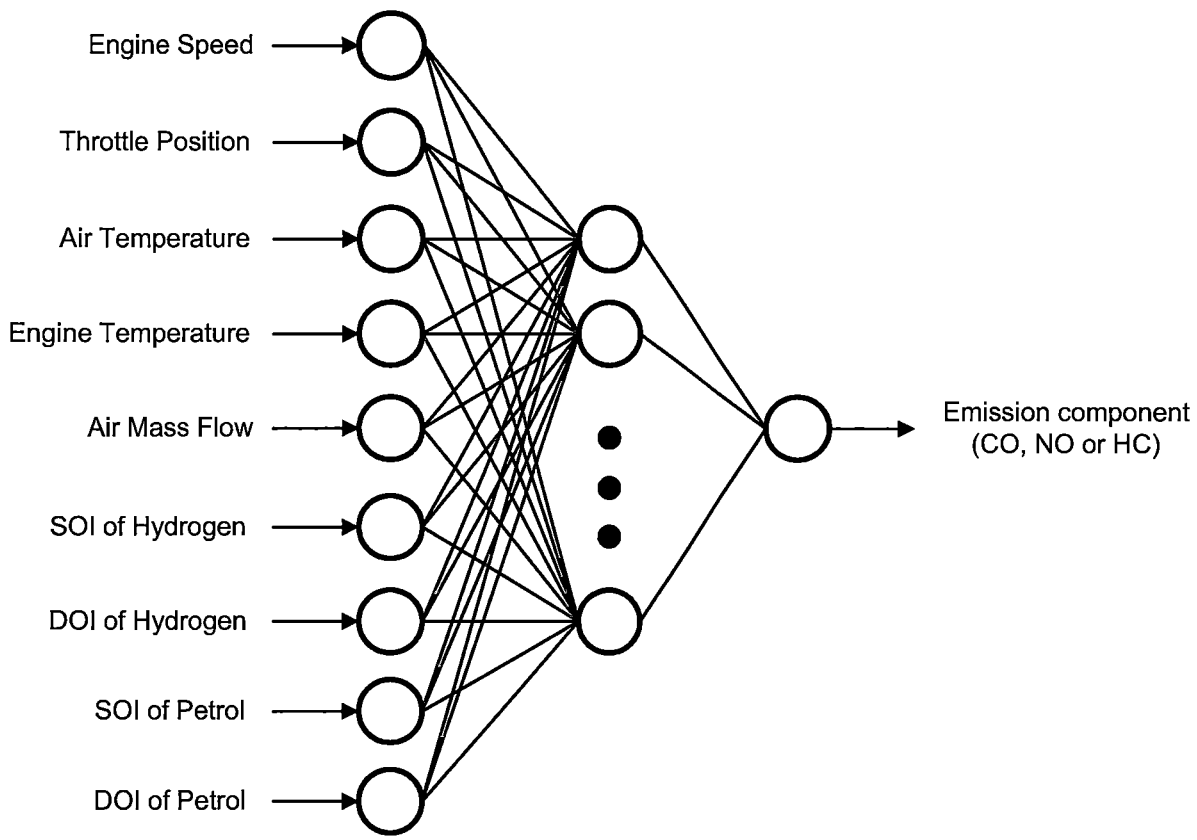


Figure 6.1: *Neural Network Predictive Model for prediction of engine emission components*

For each of the emission gas to be modeled, four types of Neural Networks described in Section 3.5.2 of Chapter 3 are applied. They are:

- Back-propagation Neural Network (1 and 2 hidden layers)
- Radial Basis Function incorporated with Kohonen Neural Network
- Optimisation Layer by Layer Neural Network

- Hybrid Neural Network

Modeling of each emission gas for prediction has been accomplished by following the steps below:

- Step 1: Data pre-processing
- Step 2: Network training and testing
- Step 3: Network selection

6.3.1. Step 1 – Data pre-processing

Before data can be used, normalisation must be applied to scale values of all data patterns to the range from 0 to 1. After being normalised, the majority of the data is used for network training and approximately 10% of the set is reserved for testing.

The data set used for modeling is comprised of 328 experimental data produced by multiple testing of the dual fuel engine at different operating conditions. The data pattern has been divided into a sub-set of 228 data used for training and a subset of 100 data for testing.

6.3.2. Step 2 – Network training and testing

For each of the network model chosen, a particular range of network architecture and settings is tested. Based on that, the network that yields minimum prediction error (RMS error) can be identified.

It is important to note that the value of RMS error easily compresses the results from all data patterns into one number without any cancellation of negative and positive errors.

For each of the emission gases the RMS error of the corresponding network being applied is plotted to assist the selection for the appropriate model.

6.3.3. Step 3 – Network selection

After being tested, network models are quantitatively benchmarked to select the one that has exhibited the best performance for a particular application. The selection criterion is the RMS error resulted from appraising a specific network model with testing data set.

6.4. NEURAL NETWORK MODELS FOR CO PREDICTION

In the first instance the Back Propagation neural networks with one hidden layer containing from 2 to 20 neurons have been tested as shown in Figure 6.2. The network with 8 hidden neurons yielding the minimum RMS error for the testing set is chosen as the optimal 1-Hidden Layer Back Propagation network for predicting CO. The RMS error values are 8.9% for the training set and 11.1% for the testing set.

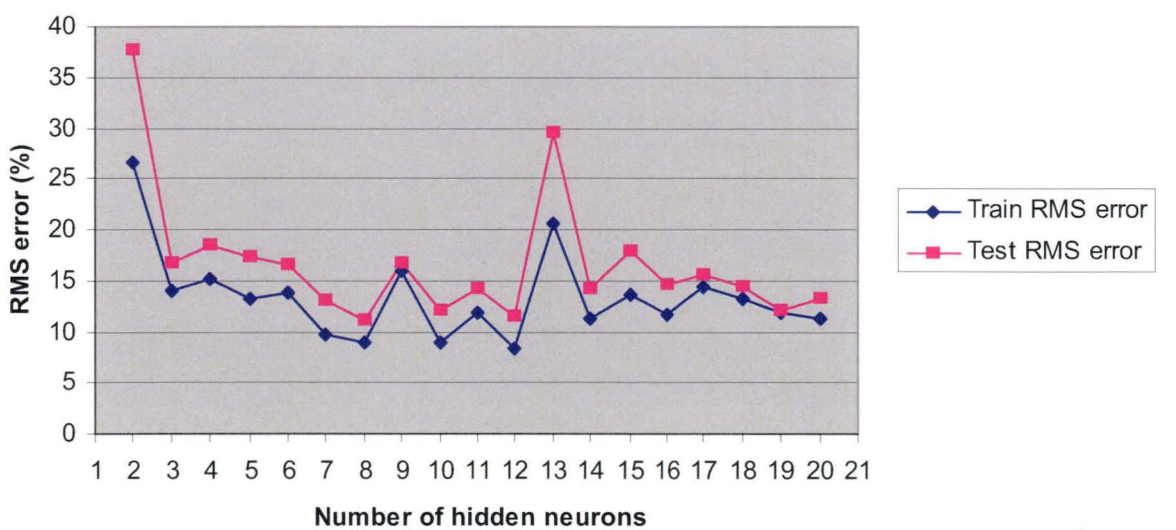
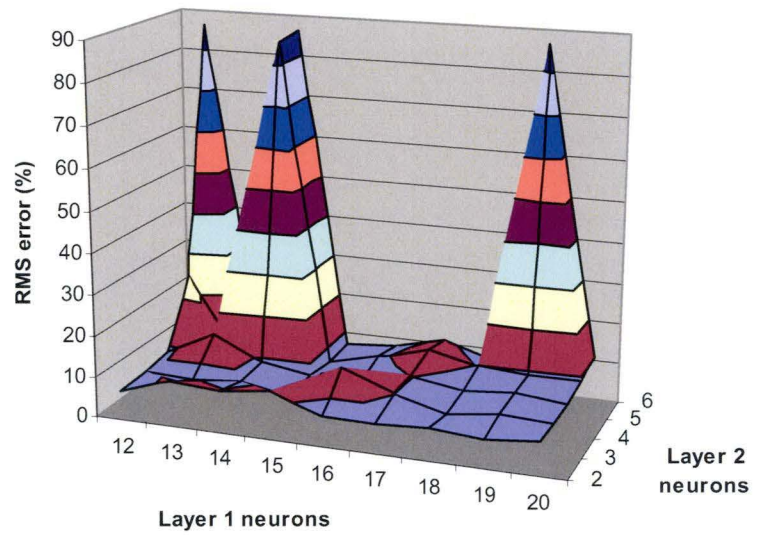
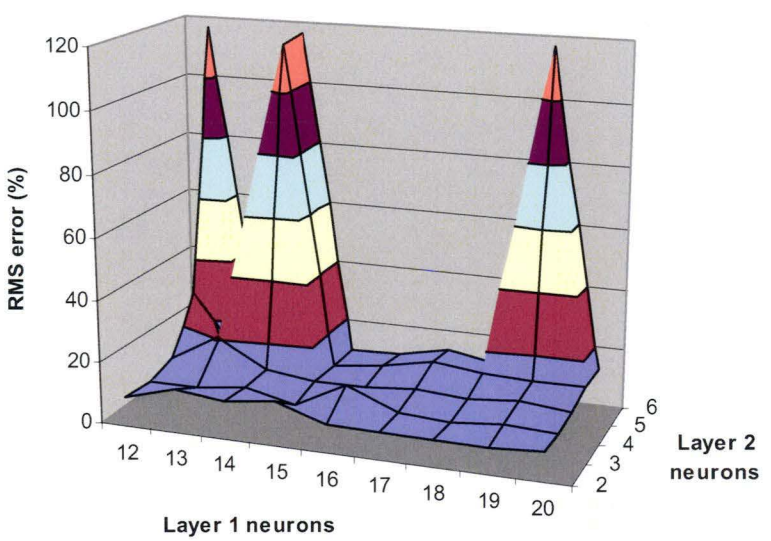


Figure 6.2: *1-Hidden Layer Back Propagation network RMS error with changing architecture for CO emission prediction*

For the 2-Hidden Layer Back Propagation neural network in Figure 6.3, the network with 15 neurons in the first hidden layer and 3 neurons in the second hidden layer has been selected. This network architecture produces RMS errors of 7.5% for the testing data set and 6.2% for the training set.



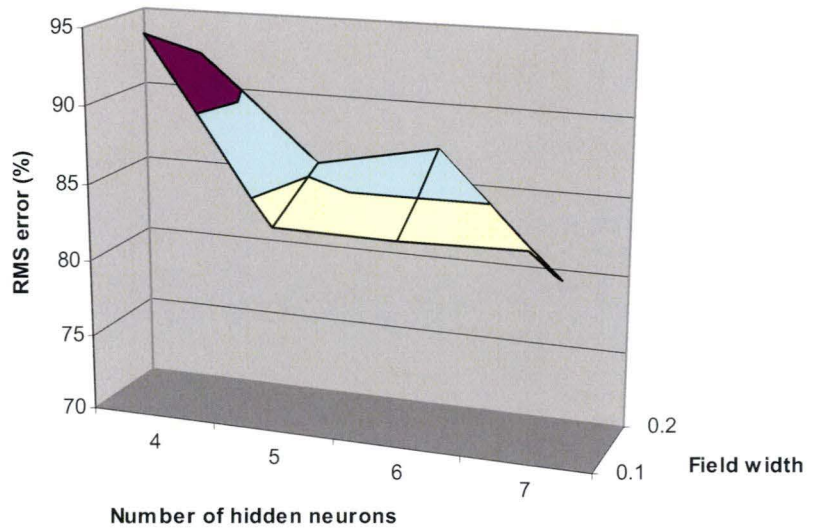
(a)



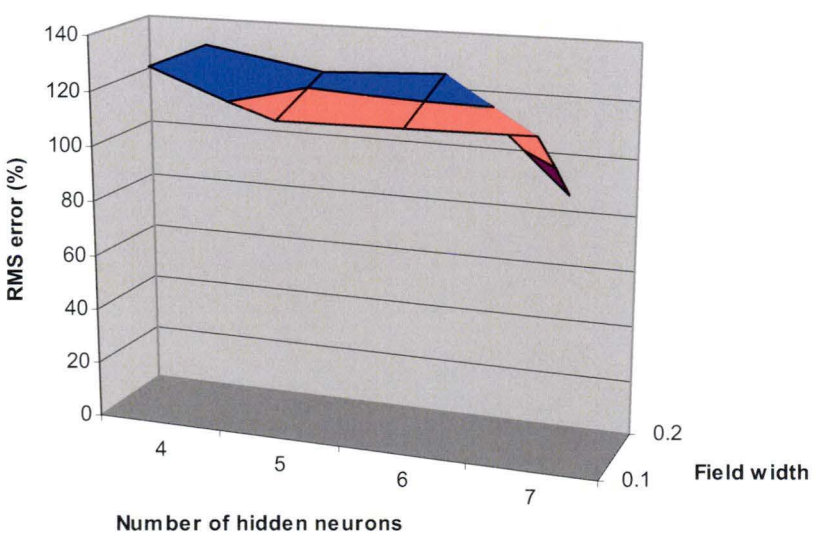
(b)

Figure 6.3: *2-Hidden Layer Back Propagation network RMS error with changing architecture for (a) Training and (b) Testing data for CO emission prediction*

In comparison with the Back Propagation network models discussed previously, the Radial Basis Function incorporating Kohonen network has proven inappropriate for this CO prediction application. This model category has been tested with the number of hidden neurons varying from 4 to 7 with receptive field widths of 0.1 and 0.2. The minimum RMS error for the testing data set that could be achieved is 85.4%. RMS error results are shown in Figure 6.4.



(a)



(b)

Figure 6.4: *Radial Basis Function incorporating Kohonen network RMS error with changing architecture for (a) Training and (b) Testing data for CO emission prediction*

As shown in Figure 6.5, the Optimisation Layer by Layer with 9 hidden neurons has been selected as the optimum of this category. It yields a minimum RMS error of 2.4% for the testing set and 2.7% for the training set.

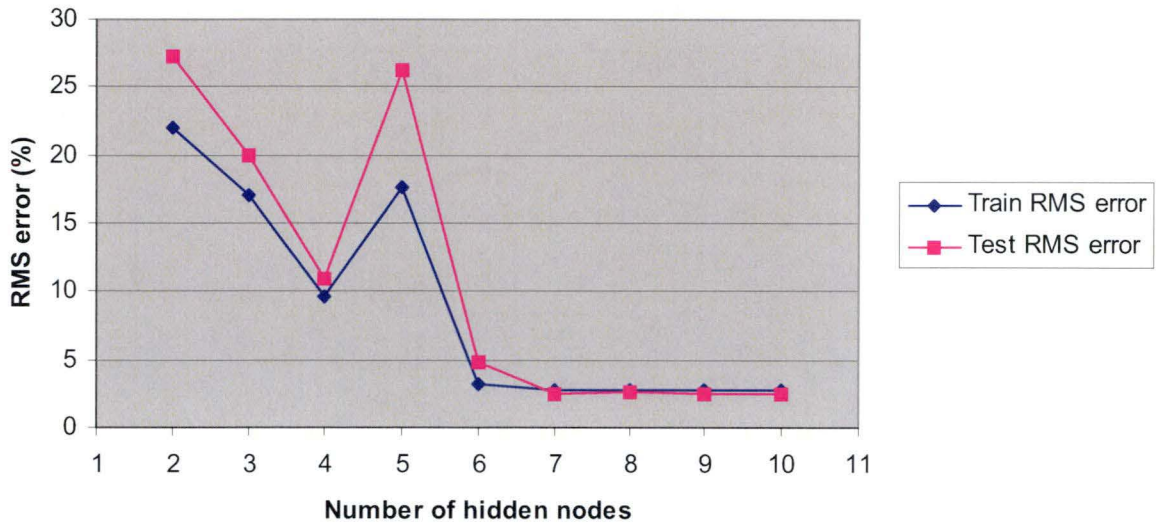
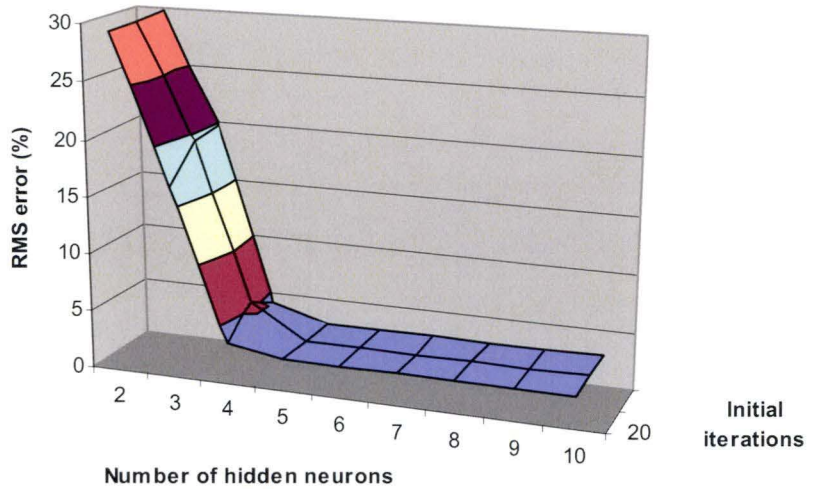
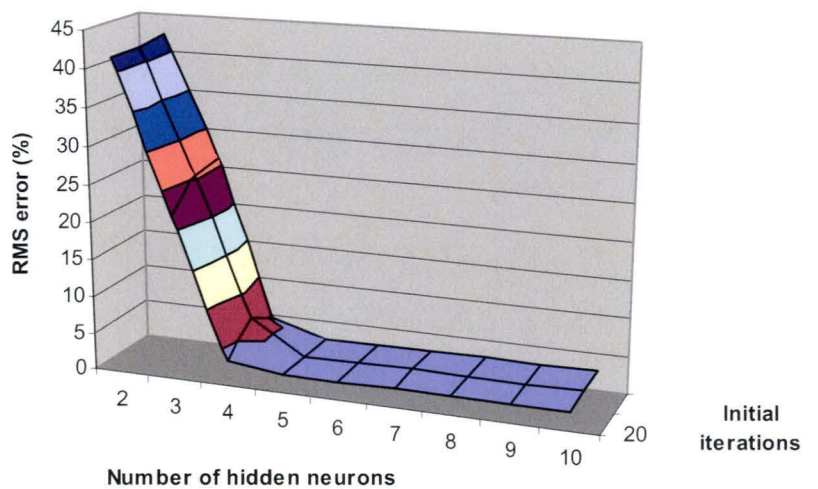


Figure 6.5: *Optimisation Layer by Layer network RMS error with changing architecture for CO emission prediction*

With the Hybrid network type, as shown in Figure 6.6, the accuracy increases with the number of hidden neurons. The architecture that consists of 5 neurons in its hidden layer and trained with 20 initial iterations has been determined to be the best model among this category. It gives resulting RMS errors of 2.5% for the testing data and 2.7% for the training data.



(a)



(b)

Figure 6.6: *Hybrid network RMS error with changing architecture for (a) Training and (b) Testing data for CO emission prediction*

In conclusion, Table 6.1 summarises the optimum models for the CO emission gas prediction application with their RMS errors and associated time taken for training to converge.

Neural Network Models	RMS error (%)		Training time (seconds)
	Training	Testing	
Back Propagation 1 hidden layer	8.9	11.1	179
Back Propagation 2 hidden layers	6.2	7.5	178
Optimisation Layer by Layer	2.7	2.4	130
Hybrid	2.7	2.5	119

Table 6.1: *Comparison of modeling results for CO emission gas prediction*

6.5. NEURAL NETWORK MODELS FOR HC PREDICTION

The Back Propagation neural networks with one hidden layer consisting from 2 to 20 neurons have been tested first. As seen in Figure 6.7, the network with 4 hidden neurons has resulted in the minimum RMS error for the testing set. Therefore it has been chosen as the optimal 1-Hidden Layer Back Propagation network for predicting HC. The RMS error values are 9.2% and 12.1% for the training set and the testing data set, respectively.

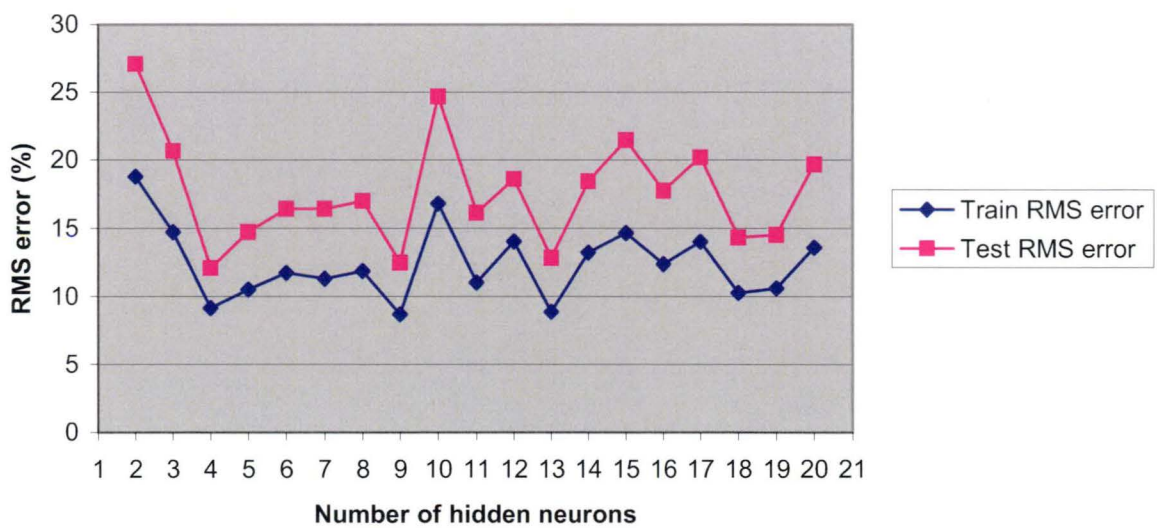
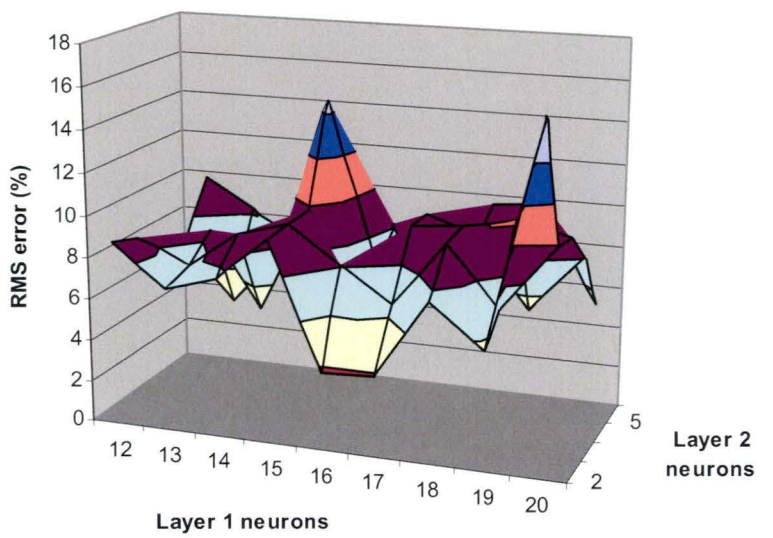
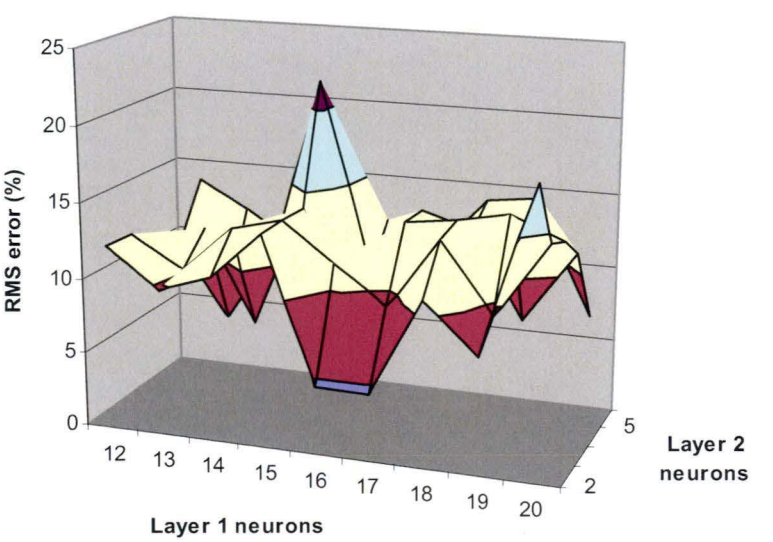


Figure 6.7: *1-Hidden Layer Back Propagation network RMS error with changing architecture for HC emission prediction*

As illustrated in Figure 6.8, the 2-Hidden Layer Back Propagation neural networks with different architectures yield different RMS error. The model with 17 neurons in the first hidden layer and 2 neurons in the second hidden layer has been selected. This network architecture has produced RMS errors of 4.5% for the testing data set and 3.9% for the training set.



(a)

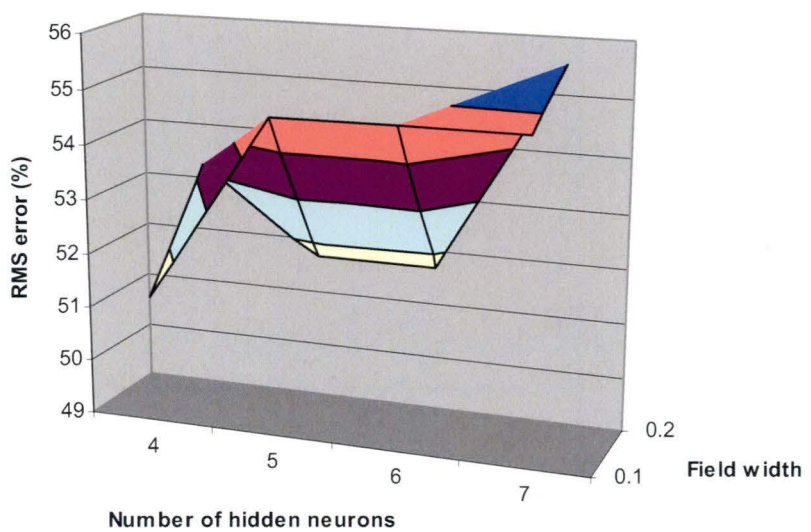


(b)

Figure 6.8: *2-Hidden Layer Back Propagation network RMS error with changing architecture for (a) Training and (b) Testing data for HC emission prediction*

In comparison with the Radial Basis Function incorporating Kohonen network used in prediction of CO, accuracy of networks of this type when used to predict HC emission has improved. This model category has been tested with the number of hidden neurons varying from 4 to 7 with receptive field widths of 0.1 and 0.2 and the minimum RMS error for the testing data set that could be achieved is 51.32%. Therefore Radial Basis Function network is found inappropriate for this particular application. RMS error results are shown in Figure 6.9.

(a)



(b)

Figure 6.9: *Radial Basis Function incorporating Kohonen network RMS error with changing architecture for (a) Training and (b) Testing data for HC emission prediction*

Based on the result shown in Figure 6.10, the Optimisation Layer by Layer with 6 neurons in the hidden layer has been chosen as the best of this category for HC emission prediction. It yields the minimum RMS error of 2.5% for the testing set and 2.3% for the training set.

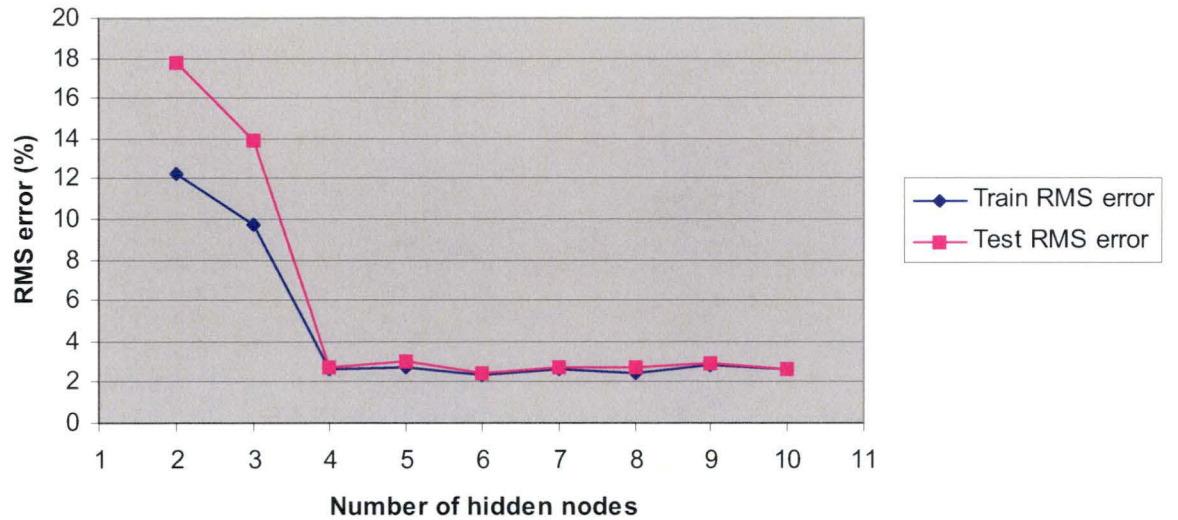
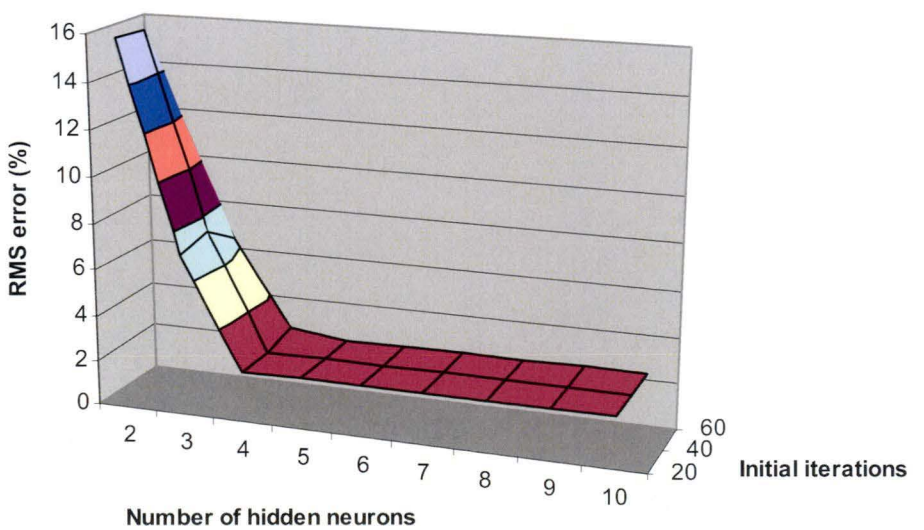
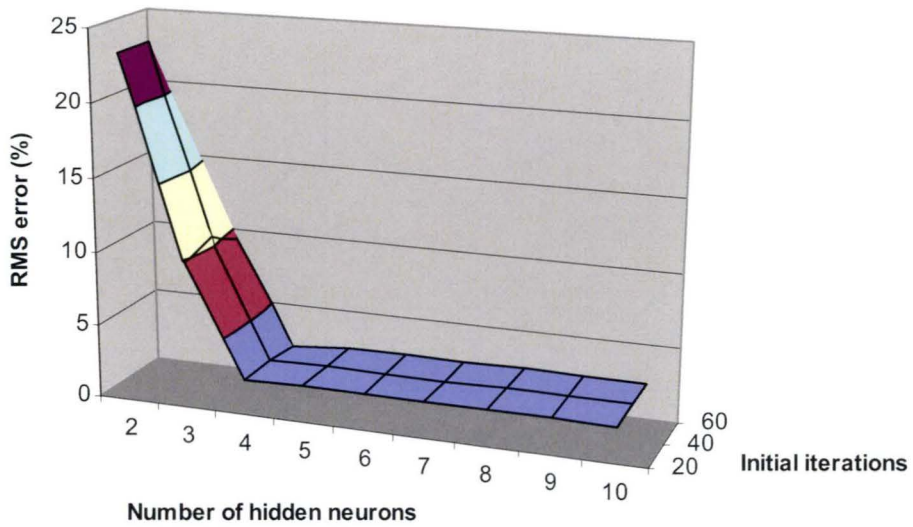


Figure 6.10: *Optimisation Layer by Layer network RMS error with changing architecture for HC emission prediction*

With the Hybrid network type, the architecture that consists of 4 neurons in its hidden layer and trained with 60 initial iterations has been selected to be the optimum model among this category. It yields RMS errors of 2.2% for the testing data and 2.6% for the training data as shown in Figure 6.11.



(a)



(b)

Figure 6.11: *Hybrid network RMS error with changing architecture for (a) Training and (b) Testing data for HC emission prediction*

In conclusion, Table 6.2 summarises the optimum models for the HC emission gas prediction application with their RMS errors and associated time taken for training to converge.

Neural Network Models	**RMS error (%)**		**Training time (seconds)**
	Training	**Testing**	
Back Propagation 1 hidden layer	9.2	12.1	173
Back Propagation 2 hidden layers	3.9	4.5	180
Optimisation Layer by Layer	2.3	2.5	100
Hybrid	2.6	2.2	55

Table 6.2: *Comparison of modeling results for HC emission gas prediction*

It can be seen that the Optimisation Layer by Layer and Hybrid networks have shown excellent training and testing results with substantially smaller training times, compared to the Back Propagation models.

6.6. NEURAL NETWORK MODELS FOR NO PREDICTION

For modeling of NO emission, the Back Propagation neural networks with one hidden layer containing from 2 to 9 neurons have been trialed as shown in Figure 6.12. When the number of hidden neurons is higher than 9, training seems to be not convergent. The network with 7 hidden neurons yielding the minimum RMS error for the testing set is chosen as the optimal 1-Hidden Layer Back Propagation network for predicting NO. The RMS error values are 4.5% for the training set and 3.6% for the testing set.

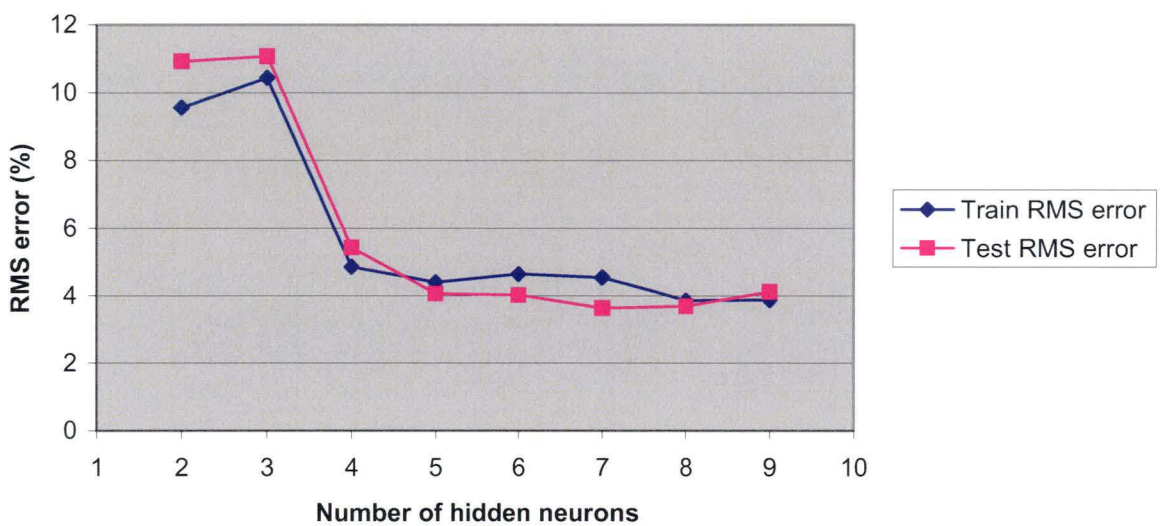
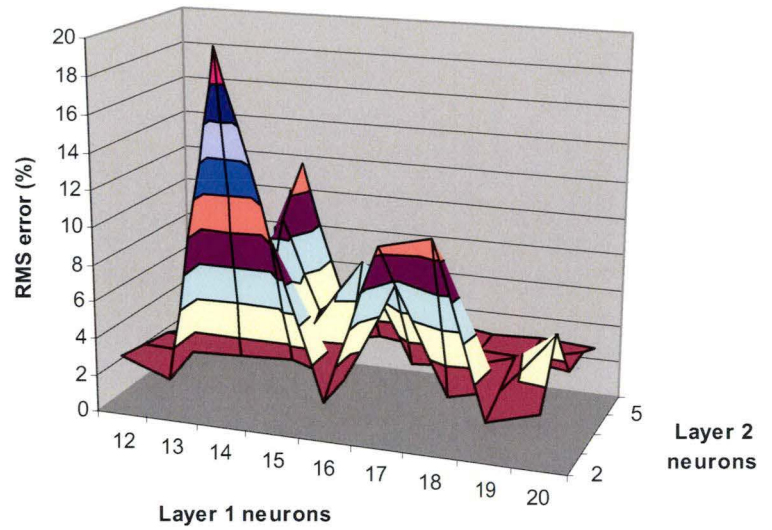
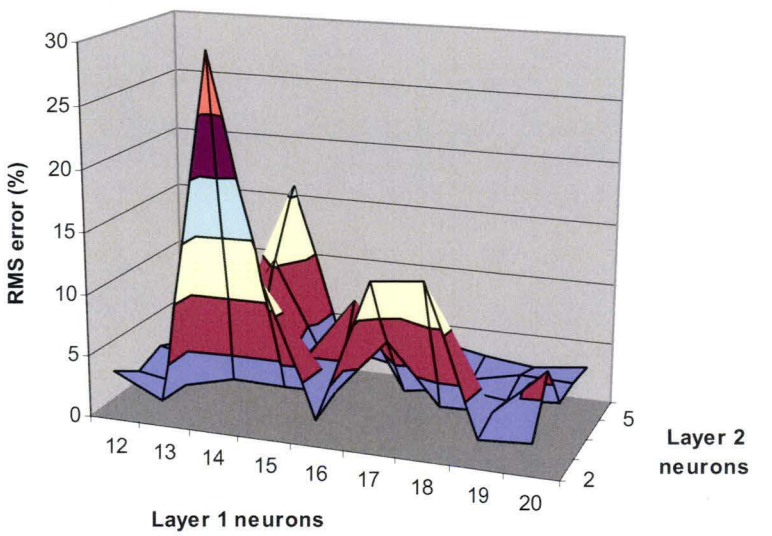


Figure 6.12: *1-Hidden Layer Back Propagation network RMS error with changing architecture for NO emission prediction*

For the 2-Hidden Layer Back Propagation neural network in Figure 6.13, the network with 13 neurons in the first hidden layer and 3 neurons in the second hidden layer has been selected. This network architecture produces an RMS error of 2% for the testing data set and 2.9% for the training set.



(a)

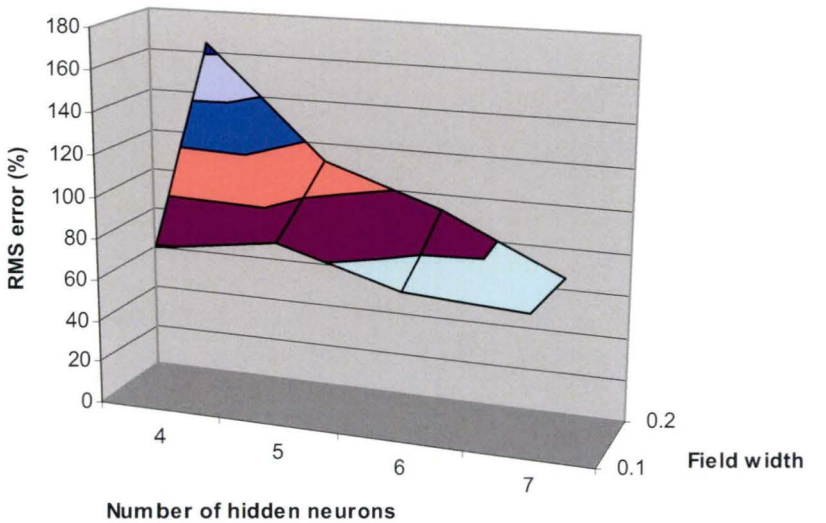


(b)

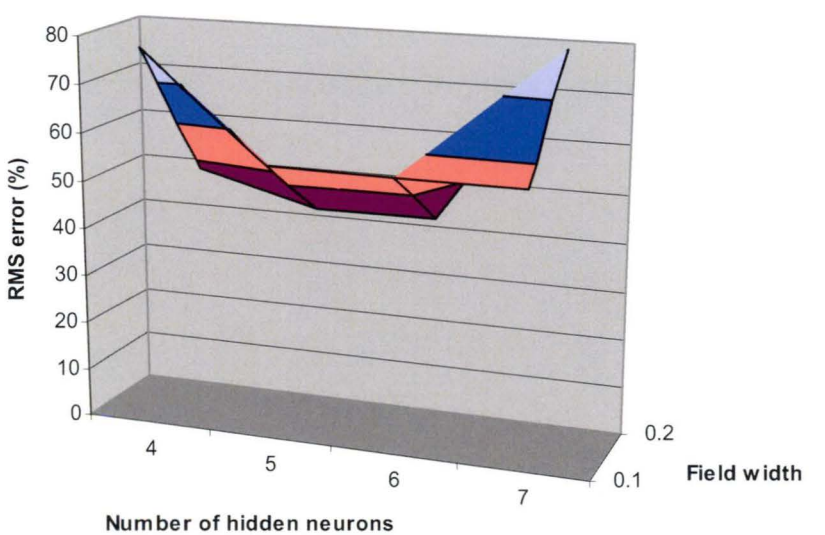
Figure 6.13: *2-Hidden Layer Back Propagation network RMS error with changing architecture for (a) Training and (b) Testing data for NO emission prediction*

Once again, the Radial Basis Function incorporating Kohonen network has shown its inadequacy in this specific emission predictive application. This model category has been tested with the number of hidden neurons varying from 4 to 7 with receptive field widths of 0.1

and 0.2. The minimum RMS error for the testing data set that could be achieved is 65.3%. RMS error results are shown in Figure 6.14.



(a)



(b)

Figure 6.14: *Radial Basis Function incorporating Kohonen network RMS error with changing architecture for (a) Training and (b) Testing data for NO emission prediction*

As shown in Figure 6.15, the Optimisation Layer by Layer with 8 hidden neurons has been selected as the optimum of this category. It yields a minimum RMS error of 4.5% for the testing set and 5.3% for the training set.

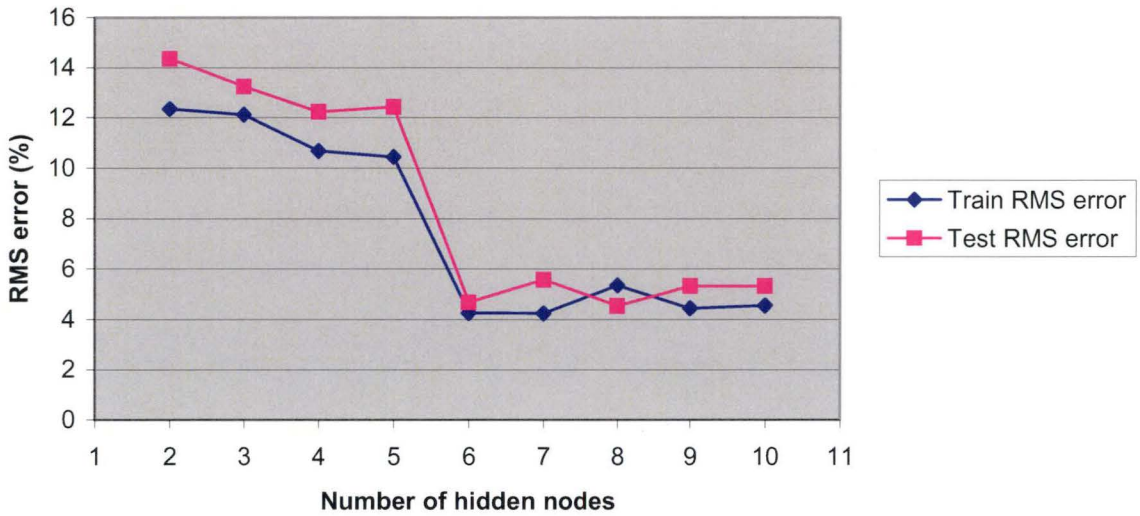
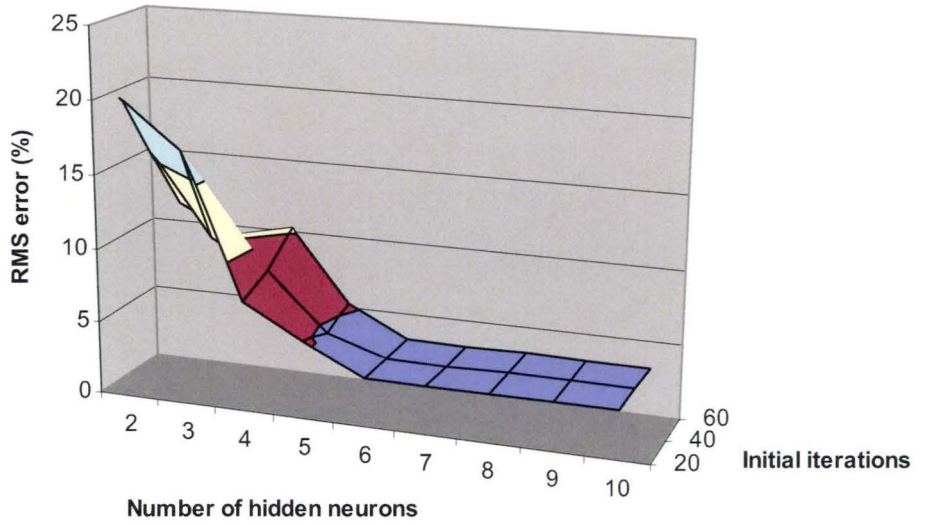
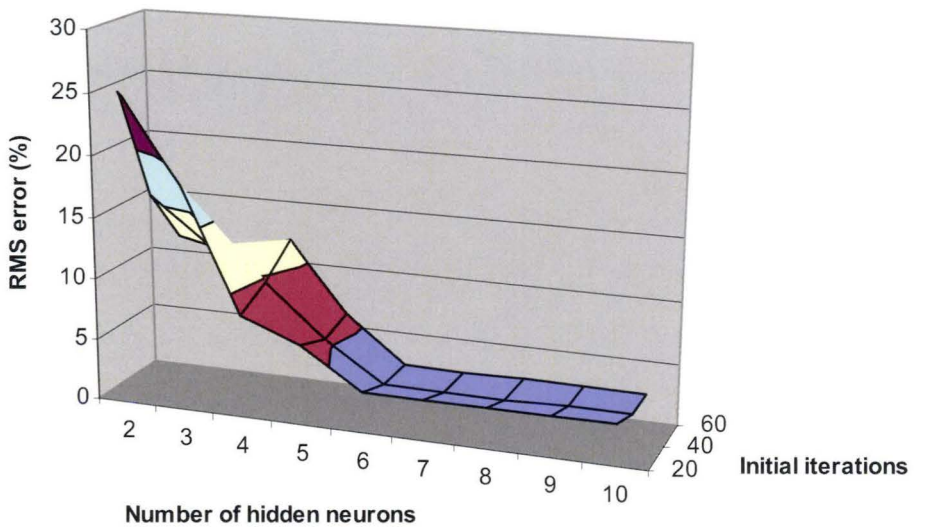


Figure 6.15: *Optimisation Layer by Layer network RMS error with changing architecture for NO emission prediction*

With the Hybrid network type, as shown in Figure 6.16, the accuracy increases with the number of hidden neurons. The architecture that consists of 6 neurons in its hidden layer and trained with 40 initial iterations has been determined to be the best model among this category. It gives a resulting RMS error of 2.3% for the testing data and 3.2% for the training data.



(a)



(b)

Figure 6.16: *Hybrid network RMS error with changing architecture for (a) Training and (b) Testing data for NO emission prediction*

In conclusion, Table 6.3 summarises the optimum models for the NO emission gas prediction application with their RMS errors and associated time taken for training to converge.

Neural Network Models	RMS error (%)		Training time (seconds)
	Training	Testing	
Back Propagation 1 hidden layer	4.5	3.6	155
Back Propagation 2 hidden layers	2.0	2.9	175
Optimisation Layer by Layer	5.3	4.5	95
Hybrid	3.2	2.3	110

Table 6.3: *Comparison of modeling results for NO emission gas prediction*

6.7. CONCLUDING REMARKS

Through study of various neural networks into prediction of three specific emission gases (CO, HC, NO), high accuracy has been achieved with some models. It gives confidence when adopting those models for online prediction.

It has been shown that for each of the gasses to be predicted, the behavior of applied neural network models with respect to varying architectures and network parameters is significantly different in RMS error and training time. Selection of the most appropriate models has also been discussed.

In CO emission modeling, the Optimisation Layer by Layer network with 9 hidden neurons and the Hybrid network with 5 hidden neurons and trained with 20 initial iterations have yielded optimum accuracy. The first one produces RMS errors of 2.4% for the testing set and 2.7% for the training set while the second one has given RMS errors of 2.5% for the testing data and 2.7% for the training data. However, in terms of the training time, Hybrid network is more preferred for this specific predictive application (119 seconds).

For HC gas prediction, the Optimisation Layer by Layer with 6 neurons in the hidden layer has yielded RMS errors of 2.5% for the testing set and 2.3% for the training set. The Hybrid network consisting 4 hidden neurons and trained with 60 initial iterations has resulted RMS errors of 2.2% for the testing data and 2.6% for the training data. It has also been shown that even those two networks have produced nearly the same RMS error, the training time of the Hybrid network is only half of that of the Optimisation Layer by Layer network (55 seconds versus 100 seconds).

With NO prediction, the Back Propagation model with 2 hidden layers and the Hybrid network are the two that have generated minimum RMS errors. The 2-Hidden Layer Back Propagation neural network with 13 neurons in the first hidden layer and 3 neurons in the second hidden layer has produced RMS errors of 2% for the testing data set and 2.9% for the training set. With Hybrid category, the network with 6 neurons in its hidden layer and trained with 40 initial iterations has propagated RMS errors of 2.3% for the testing data and 3.2% for the training data. The training time of the Hybrid network is significantly less than the training time of the Back Propagation model (110 seconds comparing with 175 seconds).

CHAPTER 7

ONLINE APPRAISAL OF THE SYSTEM

7.2. ENGINE TORQUE RESULTS

In the test the pressure of Petrol and Hydrogen fuel rail was maintained at 3bars. The flow rate of each Petrol injector was 150g/min and the flow rate of each Hydrogen injector was 200g/min. Tuning of the engine running on Petrol-H_2 mixing mode was targeted at reducing the amount of Petrol supply while keeping the torque curve as close to the Petrol curve as possible.

The graphs below show the results of testing of the engine at different engine condition.

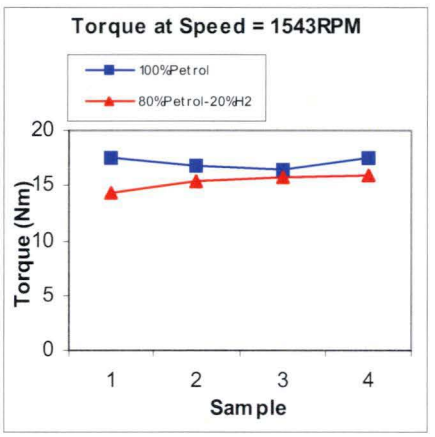


Figure 7.1: *Engine torque at 1543RPM*

Figure 7.1 shows the engine torque recorded when the engine was idling. At this operation point when running the mixing mode 20% (by mass) of Hydrogen was added and the amount of Petrol was also reduced. The power loss comparing with running the engine on 100% Petrol at this point was not significant and the torque measured was steady.

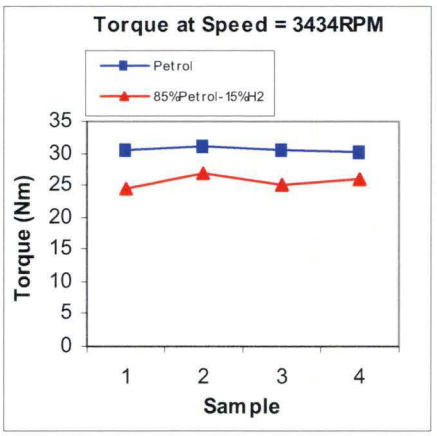


Figure 7.2: *Engine torque at 3434RPM*

At the mixing test condition of engine speed being 3434RPM (Figure 7.2) the percentage of H_2 was reduced to 15%. The tuned engine could not reach the torque when running on 100% Petrol quantitatively but the measured values were steady over the collected test samples.

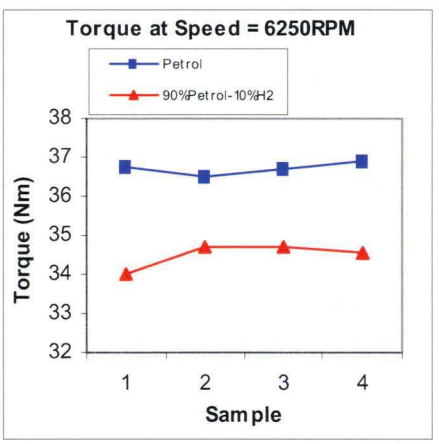


Figure 7.3: *Engine torque at 6250RPM*

At the test point of 6250RPM (Figure 7.3) the percentage of H_2 in the fuel mixture was further reduced to 10%. The power loss was approximately 7% and the measured torque at this specific condition was not highly variant.

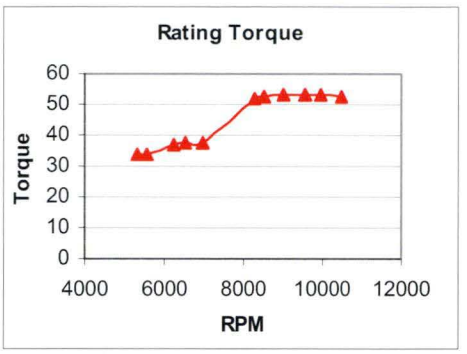


Figure 7.4: *Rating torque curve* [100]

Figure 7.4 shows the rating torque curve of this specific test engine running purely on Petrol that was well established by previous researches within the School of Engineering and is used as the reference.

The overall torque curve resulting from the dual fuel operation of the test engine is shown in Figure 7.5.

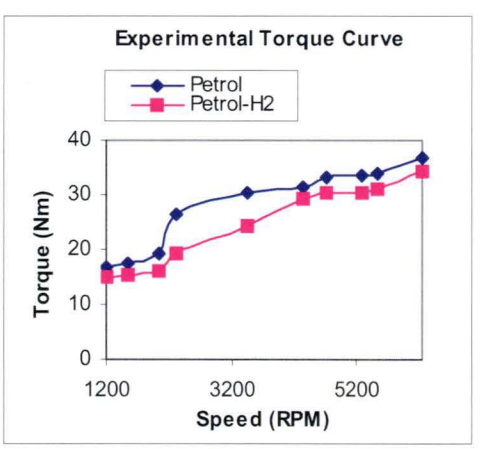


Figure 7.5: *Engine overall torque curve*

It has been shown that although the torque curve from the mixing testing is lower than that of pure Petrol operation, the trend of the newly established torque curve when running on Petrol-Hydrogen mixing mode follows the trend of the torque when running on Petrol only. It gives the confidence of the workability of the designed system.

7.3. EMISSION PREDICTION RESULTS

In order to appraise the online prediction capability of the designed system, the weight matrices generated during the offline training have been incorporated into the control code of the PC104 computer board.

For every neural network the *Neural Network Analysis Package* uses to analyse research data, it generates 2 text files called *psc.txt* and *wts.txt*. The *psc.txt* file contains information about the network architecture while the *wts.txt* file contains the weight matrix resulting from the training process of that particular network. Data in those files is then incorporated into the PC104 for online prediction.

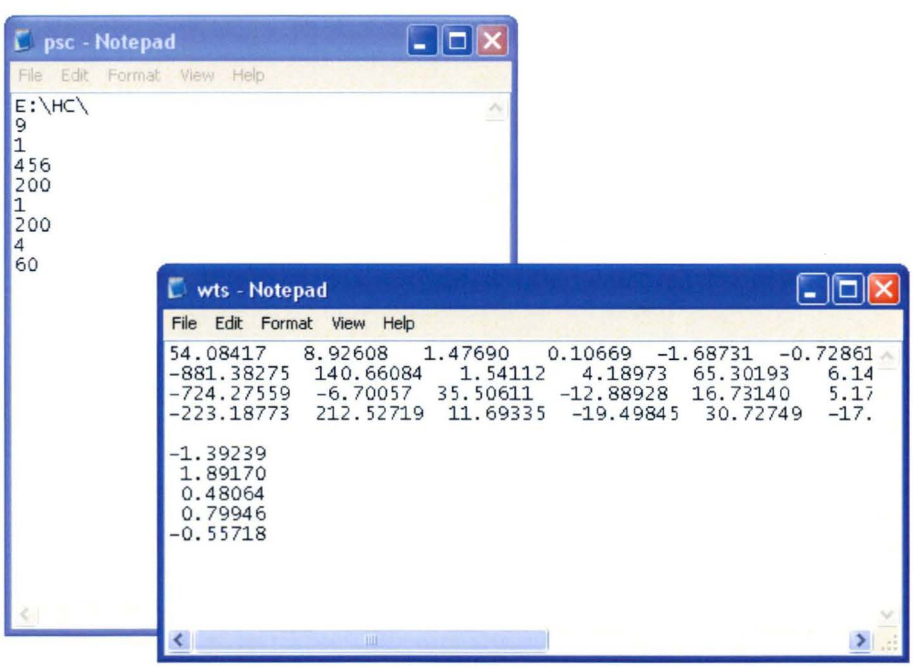


Figure 7.6: *Network configuration and weight files*

As discussed in Section 5.7.5 of Chapter 5, when the control system performs the prediction task, it actually multiplies the inputs taken from sensors and look-up tables with the weight matrices. Figure 7.7 shows the screen that displays the control parameters and prediction results when a monitor is connected to the PC104 board.

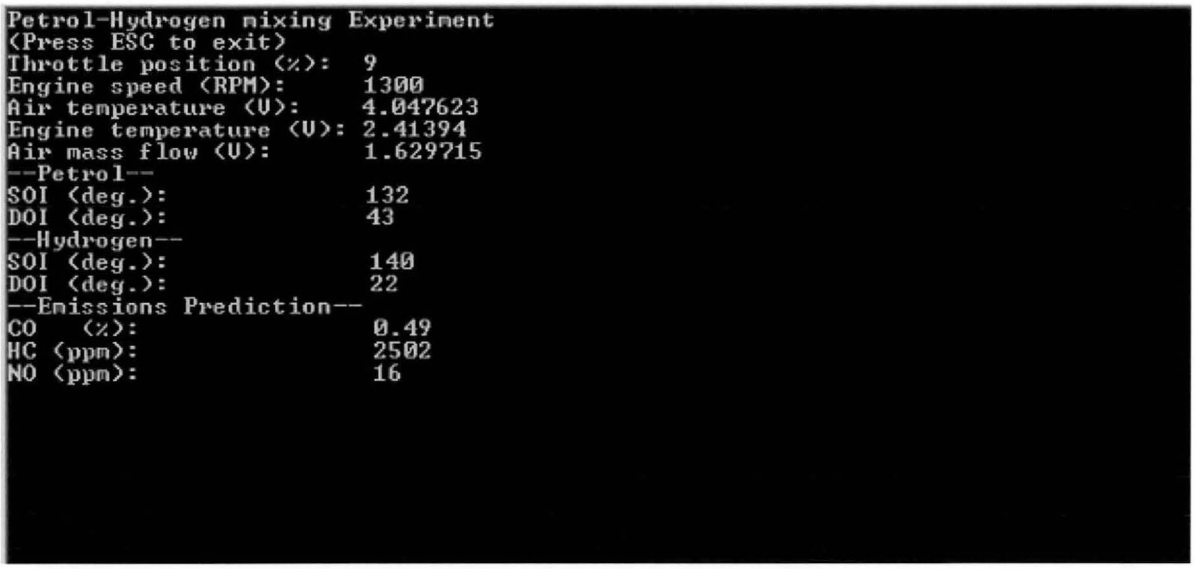


Figure 7.7: *Online display*

For online testing, the Hybrid network type has been selected for CO and NO prediction while Optimisation Layer by Layer has been chosen for HC predictive calculation. The Hybrid model for CO prediction has 5 neurons in its hidden layer and trained with 20 initial iterations while the model for NO prediction has 6 hidden neurons and trained with 40 initial iterations. The optimum Optimisation Layer by Layer model selected for HC prediction has 6 hidden neurons. The predicted values have been verified against measurement readings from the ADS 9000 Super Four Gas Analyser. Table 7.1 shows online prediction results for 6 operating conditions of the dual fuel engine. Appendix A shows the testing results in detail.

Operating condition number	**CO prediction error (%)**	**HC prediction error (%)**	**NO prediction error (%)**
1	10.9	6.1	4.8
2	15.5	18.7	12.5
3	8.4	5.8	11.8
4	15.6	11.6	5.3
5	14.6	14.3	6.7
6	17.7	8.1	3.0

Table 7.1: *Results for online emission prediction*

CHAPTER 8

FINAL CONCLUDING REMARKS AND PROPOSED FUTURE WORKS

This thesis provides comprehensive information about dual fuel systems and predictive modeling of engine emissions.

From the extensive literature survey it has been shown that much of the technology associated with IC engines is based on single fuel. This thesis has given a comprehensive overview of the Spark Ignition (SI) Internal Combustion (IC) engine structure and control strategy as well as current mechanical techniques for converting conventional engines to dual fuel engines. The need of online emission prediction in automotive engineering is highlighted. In addition to that, conventional and neural network techniques in the emission-modeling field have been summarised. They will replace expensive sensors and those mathematical models will be incorporated into modern electronic processors and operate as virtual sensors.

As part of the literature investigation, neural network has been proven to be an effective tool when applied into fields such as character recognition, signal processing and financial...Theory and mathematical logic behind each of the neural network models used in this project have also been discussed. A good understanding of these models has been demonstrated with *Neural Network Analysis Package,* a Visual Basic, Pascal based program running on Excel environment developed by researchers of the School of Engineering including the author.

Design of an embedded Hydrogen fuel injection system with associated instrumentation has been described. The conversion process of the Kawasaki 660cc engine has been discussed. Mechanical design of the dual fuel intake manifold, which is tailored-made to facilitate the electronic control instrumentation, has also been discussed. The injection control methodology is based on mapping tables and interpolation between data points. Timing pattern of injection has been depicted and analysed. From the result point of view the trend of torque when the engine runs on mixing mode follows the torque curve when it runs on 100% Petrol. This is a reassurance on the add-on injection module for mixing Hydrogen and Petrol.

Besides the control aspect of the system, prediction of emissions has also been discussed in this thesis. Various neural network models have been used for offline training of the collected data to model 3 emission gases. Those gases are CO, HC and NO.

In CO emission modeling, the Hybrid network with 5 hidden neurons and trained with 20 initial iterations have yielded optimum accuracy. It has produced RMS errors of 2.5% for the testing data and 2.7% for the training data and training time was 119 seconds.

For HC gas prediction, the Optimisation Layer by Layer with 6 neurons in the hidden layer has yielded minimum errors among the tested models with a training time of 100 seconds. Values of RMS error are 2.5% for the testing set and 2.3%.

With NO prediction, the Hybrid neural network with 6 neurons in its hidden layer and trained with 40 initial iterations has been chosen as it has propagated RMS errors of 2.3% for the testing data and 3.2% for the training data. Training time of this network model is 110 seconds.

With the network models that have been found appropriate for each particular gas, their weights have been incorporated into the control program to perform online prediction. Online prediction has shown excellent quantitative prediction with ± 10 % accuracy.

From the background set-up by this thesis a number of improvements can be suggested. Control of injection based on mapping tables can be replaced by a neural network-based algorithm. An online testing incorporating the intelligent tools embedded into microcontrollers to estimate injection and ignition timing and subsequent prediction of emissions will be beneficial. This tool is particularly relevant when the limits on emissions are imposed and that the online control of process variables is necessary to restrict the emissions within the regulated limits.

REFERENCES

[1] American Automobile Manufacturers Association (AAMA), World Motor Vehicle Data 1993 (AAMA, Washington, D.C., 1993), p. 23 and American Automobile Manufacturers association (AAMA), Motor Vehicle Fact and Figures 1996 (AAMA, Washington, D.C., 1996), p. 44.

[2] American Automobile Manufacturers Association (AAMA). "Motor Vehicle Facts and Figures 1996", (AAMA, Washington, D.C., 1996), pp. 44-47.

[3] WRI calculation based on compiled data from various editions of: International Energy Agency, Energy Statistics and Balances of Non-OECD Countries and Energy Statistics and Balances of OECD Countries (Organization for Economic Co- Operation and Development, Paris, various editions).

[4] Environment Protection Authority (EPA), Australia, "Motor Vehicle Emissions and Air Quality", http://www.epa.vic.gov.au/Air/Issues/vehicle_emissions.asp, Accessed: 28 June 2006.

[5] J.T. Houghton et al., eds., "Climate Change 1995: The Science of Climate Change", published for the Intergovernmental Panel on Climate Change, in collaboration with the World Meteorological Organization and the United Nations environment Program (Cambridge University Press, Cambridge, U.K., 1996).

[6] Bureau of Meteorology, Australian Government, "Annual Australian Climate Statement 2005",http://www.bom.gov.au/announcements/media_releases/climate/change/20060104.shtml, Accessed: 28 June 2006.

[7] Robert G. Watts, ed., "Engineering Response to Global Climate Change: Planning a Research and Development Agenda", New York: Lewis Publishers, 1997, p. 6.

[8] Bascom, R., Bromberg, P.A., Costa, D.A., Devlin, R., Dockery, D.W., Frampton,M.W.,Lambert, W., Samet, J.M., Speizer, F.E., and Utel, M. (1996). Health effects of outdoor air pollution: parts I and II, "American Journal of Respiratory Critical Care Medicine",Vol. 153, pp 477-498.

[9] Morris, R.D. and Naumova, E.N. "Carbon monoxide and hospital admissions for congestive heart failure: evidence of an increased effect at low temperatures", Environmental Health Perspectives, Vol. 106 (10), 1998, pp 649-653

[10] Streeton, J.A. "A Review of Existing Health Data on Six Air Pollutants". A report to the National Environment Protection Council, Australia. 1997.

[11] Carpenter CP, Kinkead ER, Geary DL Jr, Sullivan LJ, King JM, "Animal and human response to vapors of mixed Xylenes". Petroleum Hydrocarbon toxicity studies. V.Toxicol Appl Pharmacol, Vol. 33, 1975, pp. 543-558.

[12] Andersen I, Lundqvist GR, Molhave L, et al. "Human response to controlled levels of toluene in six-hour exposures", Scand J Work Environ Health 9, 1983, pp. 405-418.

[13] Bascom, R., Bromberg, P.A., Costa, D.A., Devlin, R., Dockery, D.W., Frampton, M.W.,Lambert, W., Samet, J.M., Speizer, F.E., and Utel, M. "Health effects of outdoor air pollution", parts I and II, American Journal of Respiratory Critical Care Medicine, Vol. 153, 1996, pp 477-498.

[14] Katsouyanni, K., Touloumi, G., Spix, C., Schwartz, J, Balbucci, F., Medina, S, Rossi, G.,Wojtyniak, B., Sunyer, J., Bacharova, L., Schouten, J.P., Ponka, A., and Anderson, H.R. "Short-term effects of ambient sulphur dioxide and particulate matter on mortality in 12 European cities: results from time series data from the APHEA project", British Medical Journal, Vol. 314 (7095), 1997, pp 1658-1663.

[15] Woodward, A., Guest, C., Steer, K., Harman, A., Scicchitano, R., Pisaniello, D., Calder, L. and McMichael, A. "Tropospheric ozone: respiratory effects and Australian air quality goals", Journal of Epidemiology and Community Health, Vol. 49, 1995, pp 401-407.

[16] McConnell, R., et al. "Asthma in exercising children exposed to ozone: A cohort study", PEDIATRICS Vol. 112 No. 2 August 2003, p. 471.

[17] Krewski, D. et al. "Reanalysis of the Harvard Six Cities Study and the American Cancer Society Study of Particulate Air Pollution and Mortality", Health Effects Institute Special Report, 2000..

[18] Lipfert, F. W. et. al. "The Washington University-EPRI veterans' cohort mortality study: preliminary results", Inhalation toxicology, 12. 2000, pp. 41-73.

[19] Hoek, G. et. al. "The association between mortality and indicators of traffic-related air pollution in a Dutch cohort study", Lancet, 360, 2002, pp. 1203-1209.

[20] Pope, C. A. et. al. "Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution", Journal of the American medical association, 287, 2002, pp. 1132-1141.

[21] Brent D. Yacobucci. "Alternative Transportation Fuels and Vehicles: Energy, Environment, and Development Issues", CRS Report for US Congress, 2005, p. 1.

[22] R.S. Benson. "Thermodynamics and Gas Dynamics of Internal Combustion Engines". Oxford University Press, 1982.

[23] R Stone. "Introduction to the Internal Combustion Engine". The Macmillan Press Ltd,1992.

[24] Principles of Automotive Vehicles, TM 9-8000, Department of the Army, Washington, DC, 1988.

[25] Ed May, William H. Crouse. "Automotive Mechanics". McGRAW-HILL. 1991.

[26] Duffy, James E., Modern Automotive Technology, Goodheart-Willcox Company Inc., South Holland, IL, 1994.

[27] Duffy, James E., Modern Automotive Mechanics, Goodheart-Willcox Company Inc., South Holland, IL, 1990.

[28] "Gasoline Engines and Selected Systems". http://www.chevron.com/products/prodserv/fuels/bulletin/motorgas/5_gasoline-engines/pg4.asp. Accessed: 24/10/2006.

[29] Sherri Singe. "Combustion engines burn super clean". Machine Design 9/1999. ProQuest Science Journal.

[30] Bui Van Ga et al. "Application of LPG on motorcycles and small sized bus".Clean Fuels and Vehicles in Vietnam, Hanoi, 13-14 May 2004.

[31] F. Avella; Automotive fuels quality: Current and future perpectives; Rivista Combustibili (Italia), Vol. 53, No1, 1999, pp. 20-36.

[32] B. Hollemans, P. De Kok; Propane, the clean fuel of the next century for light and heavy duty vehicles Windsor workshop on alternative fuels, pp. 339-359, Toronto, Canada 12-14 Jun 1995.

[33] "High pressure direct injection". http://www.westport.com/tech/hpdi.php. Accessed on 6/11/2006.

[34] "Westport injectors". http://www.westport.com/tech/enable.php#injectors. Accessed on 6/11/2006

[35] Balluchi A. et al. "Automotive engine control and hybrid systems: challenges and opportunities. Proc. IEEE, 2000, 88(7), 888-192.

[36] T Denton. "Automotive Electrical and Electronic Systems". Arnold Publishers,2000.

[37] Press Release. Delphi Energy & Engine Management Systems, Presentation to SAE,1998.

[38] Toyota Motor Sales - Technical manual.

[39] RC Turin and H.P. Geering. "Online Identification of Air Fuel Dynamics in a Sequentially Injected Engine". SAE 930857,1993.

[40] A Amstutz N Fekete and D Powell, "Model-based Air Fuel ratio control in SI Engines with a Switch Type EGO Sensor",SAE 940972, 1994.

[41] A Chevalier, C Vigild, E Hendricks, "Predicting Port Mass Flow of SI Engines in Air/Fuel ratio control Applications",SAE paper No. 2000-01-260,2000.

[42] H. Holzmann et al."Representation of 3D mappings for Automotive Control applications using Neural Networks and Fuzzy Logic". Proceedings of the 1997 IEEE International conference on Control Applications Hartford, CT October 5-7 1997.

[43] De Nicolao G. et al. "Modeling the volumetric efficiency of IC engines: parametric, non-parametric and neural techniques". Control Eng Practice, 1996, 4(10),1405-1415.

[44] Iserman R. et al. "Modeling and adaptive control of combustion engines with fast neural networks". European Symposium on Intelligent Technologies, Hybrid Systems and their Implementation on Smart Adaptive Systems, Spain 2001.

[45] Vinsonneau J. A. F. et al. "Polynomial and neural network spark ignition engine intake manifold modeling". Proceedings of 16th International Conference on Syustems Engienering, 2003, Vol. 2. pp. 718-723.

[46] Tan Y. et al. " Neural network based non-linear dynamic modeling for automotive engines". Neurocomputing. 2000. 30. 129-142.

[47] S. W. Wang et al. "Adaptive neural network model based predictive control of an internal combustion engine with a new optimization algorithm". Proceedings of the Instituation of Mechanical Engineers. Feb 2006. p. 195.

[48] Hafner M et al. "Fast Neural Networks for diesel engine control design". Control Engineering Practice 2000;8(11):1211-21.

[49] Li XQ, Yurkovich S. "Neural network based, discrete adaptive sliding mode control for idle speed regulation in UC engines". Dynamic System Measure Control. Trans ASME 2000;122(2):269-75.

[50] Park S. et al. "Feedback error learning neural networks for spark advance control using cylinder pressure". Proceeding Institute Mechanical Engineering, Part D: Automobile Engineering 2001;215(D5):625-36.

[51] Joseph, B., & Brosilow, C. "Inferential control of processes". AIChE Journal, 24, 1978, 485–509.

[52] Qin, S. J., & McAvoy, T. J. "Nonlinear PLS modeling using neural networks". Computers and Chemical Engineering, 16(4), 1992, 379–391.

[53]Radhakrishnan, V. R., & Mohamed, A. R. "Neural networks for the identification and control of blast furnace hot metal quality". Journal of Process Control, 10(6), 2000, 509.

[54] Kresta, J. V., Marlin, T. E., & MacGregor, J. F. "Development of inferential process models using PLS". Computers and Chemical Engineering, 18(7), 1994, 597–611.

[55] Park, S., & Han, C. "A nonlinear soft sensor based on multivariate smoothing procedure for quality estimation in distillation columns". Computers and Chemical Engineering, 24, 2000, 871–877.

[56] Pitsch, H. and Barths, H. and Peters, N. "Three-Dimensional Modeling of NOx and Soot Formation in DI-Diesel Engines Using Detailed Chemistry Based on the Interactive Flamelet Approach". SAE paper 962057 (1996).

[57] Dodge, L.G., Leone, D.M., Naegeli, D.W., Dickey, D.W., and Swenson, K.R., "A PC Based Model for Predicting NOx Reductions in Diesel Engines," SAE paper 962060, 1996.

[58] Bazari A. "Diesel exhaust emissions prediction under transient operating conditions". SAE 940666 (1994).

[59] Thompson GJ, Atkinson CM, Clark NN, Long TW, Hanzevack E. "Neural Network modeling of the emissions and performance of a heavy-duty diesel engine". Proc Inst Mech Engnr, Part D: J Automobile Engng 2001; 214(D2): p. 111-26.

[60] Henrike C. Krijnsen et al. "Prediction of NOx emissions from a transiently operating Diesel engine using an Artificial Neural Network". CHEMICAL ENGINEERING & TECHNOLOGY, 22(7), 1999, pp. 601 - 607.

[61] Michael L. Traver Richartd J. Atkinson and Christopher M. Atkinson. "Neural network-based Diesel Engine Emissions Prediction Using In-Cylinder Combustion Pressure". SAE Technical Paper 1999-01-1532. (Reprinted from: Emission Formation Processes in SI and Diesel Engines SP-1462)

[62] Liu Zhen-tao, Fei Shao-mei. "Study of CNG/Diesel dual fuel engine's emissions by means of RBF neural network". Journal of Zhejiang University SCIENCE 2004 5(8):p.960-965.

[63] Caudill, M. and Butler, C. "Naturally Intelligent Systems", vol.1 Massachusetts Institute of Technology, 1990

[64] Principe, J.C. "Artificial Neural Networks" The Electrical Engineering Handbook Ed. Richard C. Dorf Boca Raton: CRC Press LLC, 2000

[65] Caudill, M. and Butler, C. "Understanding Neural Networks – Computer Explorations", vol.1 Massachusetts Institute of Technology, 1992.

[66] James A. Freeman, David M. Skapura. "Neural Networks Algorithms, Applications and Programming Techniques". Addison-Wesley Publishing Company. ISBN 0-201-51376-5

[67] Veelenturf, L. P. J. "Analysis and application of artificial neural networks". Prentice Hall. ISBN 0-13-489832-X.

[68] Negnevitsky, M. "Artificial Intelligence – A guide to Intelligent Systems". ISBN 0-201-71159-1. Pearson Education.

[69] Jurada, J. M. "Introduction to Artificial Neural Systems". West Publishing Company, 1992

[70] Hebb, D. O. "Organization of Behavior". Wiley, 1949.

[71] Avalo, E. and NAIM, P. "Neural Networks". The MacMillan Press Limited,1991.

[72] Zurada, J., "Introduction to Artificial Neural Systems", West Publishing Co,St. Paul, MN, 1992.

[73] Haykin, S., "Neural Networks: A Comprehensive Foundation", Prentice Hall, Upper Saddle River, NJ, 1999.

[74] J. Hertz, A. Krogh and R. G. Palmer. "Introduction to the Theory of Neural Computation". Addison Wesley, Reading, Mass., 1991.

[75] James A. Freeman and David M. Skapura. "Neural networks: algorithms, applications, and programming techniques". ISBN 0-201-51376-5. Addison Wesley.

[76] Hopfield, J. J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proc. Netl. Acad. Sci., vol. 79, 1982, pp. 2554-2558.

[77] Widrow, B. and Lehr, M. A., "Thirty Years of Adaptive Neural Networks: Perceptron, MADALINE, and back propagation". Proc. of the IEEE, Vol. 78, 1415-1442, 1990.

[78] Rumelhart, D.E., Hinton G.E. and Williams, R.J., "Learning Internal Representations by Error Propagation, Parallel Data Processing", Vol. 1, Chap. 8, the MIT Press, Cambridge, MA, 1986.

[79] Specht, D.F., "A General Regression Neural Network", IEEE Trans. On Neural Networks, Vol. 2, 568–576, 1991.

[80] Wasserman P.D., "Advanced Methods in Neural Computing", Van Nostrand Reinhold, New York, 1993.

[81] Moody, J. and Darken, C., "Fast Learning in Networks of Locally-Tuned Processing Units", Neural Computation, Vol. 1, 281–294, 1989.

[82] Jang, J. S., Sun, C. T., and Mizutani, E., "Neuro-Fuzzy and Soft Computing", Prentice Hall, Englewood Cliffs, NJ, 1997.

[83] Lowe, D., "Adaptive Radial Basis Function Nonlinearities and the Problem of Generalization", Proc. First IEEE Int. Conf. on Artificial Networks, London, UK, 1989.

[84] Wettschereck D. and Dietterich, T., "Improving the Performance of Radial Basis Function Networks by Learning Center Locations", Advances in Neural Information Processing Systems, Vol. 4, 1133-1140, Morgan Kaufmann, San Mateo, CA, 1992.

[85] Haykin, S., "Neural Networks: A Comprehensive Foundation", Prentice Hall, Upper Saddle River, NJ, 1999.

[86] T. Kohonen, "Self Organisation and Associative Memory". Third Edition, Springer-Verlag, New York, 1989.

[87] Haykin, S. "Neural Networks: A Comprehensive Foundation". McMillan College Publishing Company, Inc.,1994

[88] Rojas, R. " Neural Networks: A Systematic Introduction". Springer-Verlag Berlin Heidelberg, 1996.

[89] Fausett, L., "Fundamentals of Neural Networks: Architectures, Algorithms and Applications", Prentice Hall International, Inc., 1994.

[90] Kandil, N. Khorasani, K., Patel, R. V. Sood, V. K., "Optimum Learning Rate for Backpropagation Neural Networks", Neural Network: Theory, Technology and Applications. IEEE, 1996.

[91] Dayhoff, J. E. "Neural Network Architectures – An Introduction", Van Nostrand Reinhold, 1990.

[92] Brunelli, R., "Training Neural Nets Through Stochastic Minimisation". Journal Neural Networks, vol. 7, no. 9, 1994, pp.1405-1412

[93] Ben Krose, Patrick van der Smagt. "An Introduction to Neural Networks". University of Amsterdam, 1996.

[94] Huang. S. J. Zhang, H. C., "Artificial Neural Networks in Manufacturing: Concepts, Applications and Perspectives", IEEE Transactions on Components, Packaging and Manufacturing Technology, part A, vol.17, no.2, 1994, pp. 212-228

[95] Yousefizadeh, H. Zilouchian, A. "Intelligent Control Systems Using Soft Computing Methodologies". CRC Press LLC, 2001.

[96] Ergezinger, S. Thomsen, E., "An Accelerated Learning Algorithm for Multilayer Perceptrons: Optimisation Layer by Layer", IEEE Transactions on Neural Networks, vol. 6, 1995, pp. 31-42.

[97] Kiatcharoenpo, T. Karri, V.,"An Artificial Neural Network-Based Detection of Wear States in Drilling Operations", Proc. Fourth International Conference on Intelligent Technologies (InTech'03), Thailand, December 2003.

[98] Widrow, B., "An Adaptive 'Adaline' Neuron Using Chemical 'Memistors'", Technical Report Number 1553-2, Stanford Electronics Laboratories, October 1960.

[99] "Products & Services: Insurance Service Bureau". http://www.fairisaac.com/fic/en/product-service/product-index/insurance-service-bureau/ Accessed on 6/11/2006.

[100] David Frankcomb. "Internal Combustion Engine Performance Improvement". Honours Thesis 2003, University of Tasmania.

APPENDIX A
ONLINE TESTING RESULTS

RPM	Throttle %	Air temp (^{0}C)	Engine temp (^{0}C)	Air mass flow (g/s)	SOI Hydrogen (angle)	DOI Hydrogen (angle)	SOI Petrol (angle)	DOI Petrol (angle)	CO (%)	CO' (%)	Error CO (%)
1590	9	18	53	4.32	120	23	124	51	0.402	0.45	10.9
1300	9	14	44	4.46	140	22	132	43	0.414	0.49	15.5
1640	9	18	48	4.47	120	23	123	52	0.394	0.43	8.4
1710	9	14	37	4.48	120	23	117	58	0.487	0.58	15.6
3200	14	23	64	14.11	40	43	56	109	0.421	0.49	14.6
2370	16	31	62	16.18	90	24	80	90	0.418	0 51	10.9

RPM	Throttle %	Air temp (^{0}C)	Engine temp (^{0}C)	Air mass flow (g/s)	SOI Hydrogen (angle)	DOI Hydrogen (angle)	SOI Petrol (angle)	DOI Petrol (angle)	HC (ppm)	HC' (ppm)	Error HC (%)
1590	9	18	53	4 32	120	23	124	51	2794	2974	6.1
1300	9	14	44	4.46	140	22	132	43	2033	2502	18.7
1640	9	18	48	4.47	120	23	123	52	2943	3124	5.8
1710	9	14	37	4.48	120	23	117	58	2303	2604	11.6
3200	14	23	64	14.11	40	43	56	109	1202	1402	14.3
2370	16	31	62	16.18	90	24	80	90	1197	1302	8.1

RPM	Throttle %	Air temp (^{0}C)	Engine temp (^{0}C)	Air mass flow (g/s)	SOI Hydrogen (angle)	DOI Hydrogen (angle)	SOI Petrol (angle)	DOI Petrol (angle)	NO (ppm)	NO' (ppm)	Error NO (%)
1590	9	18	53	4.32	120	23	124	51	22	21	4.8
1300	9	14	44	4.46	140	22	132	43	14	16	12.5
1640	9	18	48	4.47	120	23	123	52	19	17	11.8
1710	9	14	37	4.48	120	23	117	58	18	19	5.3
3200	14	23	64	14.11	40	43	56	109	48	45	6.7
2370	16	31	62	16.18	90	24	80	90	32	33	3.0

Note: CO', HC' and NO' are predicted values.

APPENDIX B
SENSOR SPECIFICATIONS

B –1 Hall-effect Sensor

B –2 Throttle Position Sensor

B –3 Air Temperature Sensor

B –4 Engine Temperature Sensor

B –5 Air Mass Flow Sensor

A1381, A1382, A1383, and A1384

Programmable Linear Hall Effect Sensors with Analog Output Available in a Miniature Thin Profile Surface Mount Package

Features and Benefits

- Customer programmable offset, sensitivity, sensitivity temperature coefficient, and polarity
- Programmability at end-of-line
- Ratiometric sensitivity, quiescent voltage output, and clamps for interfacing with application DAC
- Temperature-stable quiescent voltage output and sensitivity
- Precise recoverability after temperature cycling
- Output voltage clamps provide short circuit diagnostic capabilities
- Wide ambient temperature range: –40°C to 150°C
- Immune to mechanical stress
- Miniature package options

Packages

3 pin surface mount
SOT23-W (suffix LH)

3 pin ultramini SIP
(suffix UA)

Not to scale

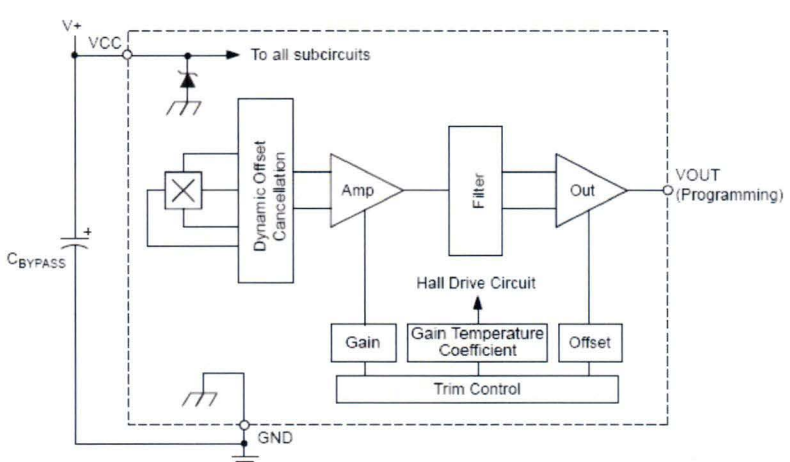
Description

New applications for linear output Hall effect sensors, such as displacement, angular position, and current measurement, require high accuracy in conjunction with small package size. The Allegro® A138x family of programmable linear Hall effect sensors was designed specifically to achieve both goals. These temperature-stable devices are available in a miniature surface mount package (SOT23-W) and an ultramini through-hole single-in-line package. The accuracy of these devices is enhanced via programmability on the output pin for end-of-line optimization without the added complexity and cost of a fully programmable device.

These ratiometric Hall effect sensors provide a voltage output that is proportional to the applied magnetic field. Both the quiescent voltage output and magnetic sensitivity are user-adjustable. The quiescent voltage output can be set around 50% of the supply voltage, and the sensitivity adjusted between 2 mV/G and 9 mV/G over the device family. Programming selections also exist for output polarity and temperature compensation. The features of this linear family make it ideal for high accuracy requirements of automotive and industrial applications, and performance is guaranteed over an extended temperature range, –40°C to 150°C.

Continued on the next page...

Functional Block Diagram



Description (continued)

Each BiCMOS monolithic circuit integrates a Hall element, temperature-compensating circuitry to reduce the intrinsic sensitivity drift of the Hall element, a small-signal high-gain amplifier, a clamped low-impedance output stage, and a proprietary dynamic offset cancellation technique.

The A138x sensors are provided in a 3 pin ultramini single-in-line package (UA suffix), and a 3 pin surface mount SOT-23W package (LH suffix).

Selection Guide

Part Number	Packing*	Package	T_A (°C)	Internal Bandwidth (kHz)	Sensitivity Range (mV/G)
A1381ELHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 85	12	6.00 to 9.00
A1381EUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1381EUATI-T	Tape and reel, 2000 pieces/reel				
A1381LLHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 150		
A1381LUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1381LUATI-T	Tape and reel, 2000 pieces/reel				
A1382ELHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 85	17	4.00 to 6.25
A1382EUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1382EUATI-T	Tape and reel, 2000 pieces/reel				
A1382LLHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 150		
A1382LUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1382LUATI-T	Tape and reel, 2000 pieces/reel				
A1383ELHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 85	21	2.75 to 4.25
A1383EUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1383EUATI-T	Tape and reel, 2000 pieces/reel				
A1383LLHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 150		
A1383LUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1383LUATI-T	Tape and reel, 2000 pieces/reel				
A1384ELHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 85	27	2.00 to 3.00
A1384EUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1384EUATI-T	Tape and reel, 2000 pieces/reel				
A1384LLHLT-T	Tape and reel, 3000 pieces/reel	Surface mount	–40 to 150		
A1384LUA-T	Bulk bag, 500 pieces/bag	Through hole			
A1384LUATI-T	Tape and reel, 2000 pieces/reel				

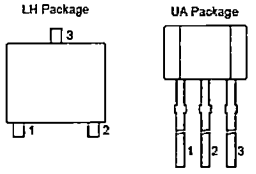
*Contact Allegro for additional packing options

Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Forward Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Forward Output Voltage	V_{OUT}		28	V
Reverse Output Voltage	V_{ROUT}		–0.1	V
Output Source Current	$I_{OUT(SOURCE)}$	VOUT to GND	2	mA
Output Sink Current	$I_{OUT(SINK)}$	VCC to VOUT	10	mA
Operating Ambient Temperature	T_A	Range E	–40 to 85	°C
		Range L	–40 to 150	°C
Storage Temperature	T_{stg}		–65 to 165	°C
Maximum Junction Temperature	T_J(max)		165	°C

Pb

Pin-out Diagrams



Number		Name	Description
LH	UA		
1	1	VCC	Input power supply; use bypass capacitor to connect to ground
3	2	GND	Ground
2	3	VOUT	Output signal, also used for programming

OPERATING CHARACTERISTICS, valid over full operating temperature range, T_A; C_{BYPASS}= 0.1 µF, V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions		Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS							
Supply Voltage	V_{CC}			4 5	5.0	5.5	V
Supply Current	I_{CC}	No load on VOUT		–	6.9	8	mA
Power-On Time[1]	t_{PO}	A1381	T_A = 25 °C, C_{BYPASS} = open, C_L (of test probe) = 10 pF, Sens = 7.5 mV/G	–	32	–	µs
		A1382	T_A = 25 °C, C_{BYPASS} = open, C_L (of test probe) = 10 pF, Sens = 5 0 mV/G	–	27	–	µs
		A1383	T_A = 25 °C, C_{BYPASS} = open, C_L (of test probe) = 10 pF, Sens = 3 125 mV/G	–	23	–	µs
		A1384	T_A = 25 °C, C_{BYPASS} = open, C_L (of test probe) = 10 pF, Sens = 2 5 mV/G	–	19	–	µs
Delay to Clamp[1]	t_{CLP}	T_A = 25°C, C_L = 10 nF		–	30	–	µs
Supply Zener Clamp Voltage	V_Z	T_A = 25°C, I_{CC} = 11 mA		6	8.3	–	V
Internal Bandwidth	BW_i	A1381	Small signal –3 dB	–	12	–	kHz
		A1382		–	17	–	kHz
		A1383		–	21	–	kHz
		A1384		–	27	–	kHz
Chopping Frequency[2]	f_C	T_A = 25°C		–	170	–	kHz
OUTPUT CHARACTERISTICS							
Noise (peak to peak)	$V_{N(p-p)}$	A1381	T_A=25°C, C_L = 10 nF, Sens = 7.5 mV/G, no external filter	–	34	–	mV
		A1382	T_A=25°C, C_L = 10 nF, Sens = 5 0 mV/G, no external filter	–	27	–	mV
		A1383	T_A=25°C, C_L = 10 nF, Sens = 3.125 mV/G; no external filter	–	20	–	mV
		A1384	T_A=25°C; C_L = 10 nF, Sens = 2 5 mV/G, no external filter	–	18	–	mV
		A138x	T_A=25°C, Sens = 2.5 mV/G, external 2 kHz low pass filter with R = 1.69 kΩ, C = 47 nF	–	4.7	–	mV
DC Output Resistance	R_{OUT}			–	< 1	–	Ω
Output Load Resistance	R_L	VOUT to VCC		4.7	–	–	kΩ
		VOUT to GND		4.7	–	–	kΩ
Output Load Capacitance	C_L	VOUT to GND		–	–	10	nF
Phase Shift[3]	ΔΦ	No load on VOUT, magnetic input signal frequency = 1 kHz, with 1 $V_{(p-p)}$ output signal		–	3	–	deg
Output Voltage Clamp[4]	$V_{CLP(HIGH)}$	T_A = 25°C, B = 600 G, Sens = 5.0 mV/G, R_L = 10 kΩ (VOUT to GND)		4.35	4.5	4 65	V
	$V_{CLP(LOW)}$	T_A = 25°C, B = –600 G, Sens = 5.0 mV/G, R_L = 10 kΩ (VCC to VOUT)		0.40	0.55	0 70	V
Output Slew Rate	SR	C_L = 10 nF		–	175	–	V/ms

OPERATING CHARACTERISTICS (continued), valid over full operating temperature range, T_A; C_{BYPASS}= 0.1 µF, V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions		Min.	Typ.	Max.	Units
PRE-PROGRAMMING TARGET[5]							
Pre-Programming Quiescent Voltage Output	$V_{OUT(Q)init}$	B = 0 G, T_A = 25°C		–	2.1	–	V
Pre-Programming Sensitivity	$Sens_{init}$	A1381	T_A = 25°C	–	4.2	–	mV/G
		A1382		–	2.9	–	mV/G
		A1383		–	2.1	–	mV/G
		A1384		–	1.4	–	mV/G
Pre-Programming Sensitivity Temperature Coefficient[6]	$TC_{Sensinit}$	T_A = 150°C		–	–0.05	–	%/°C
QUIESCENT VOLTAGE OUTPUT PROGRAMMING							
Guaranteed Quiescent Voltage Output Range[4,7]	$V_{OUT(Q)}$	B = 0 G, T_A = 25°C		2.3	–	2.6	V
Quiescent Voltage Output Programming Bits				–	6	–	bit
Average Quiescent Voltage Output Step Size[8,9]	$Step_{VOUT(Q)}$	T_A = 25°C		8	11.5	15	mV
Quiescent Voltage Output Programming Resolution[10]	$Err_{PGVOUT(Q)}$	T_A = 25°C		–	$Step_{VOUT(Q)}$ × ±0.5	–	mV
SENSITIVITY PROGRAMMING							
Guaranteed Sensitivity Range[4,11]	Sens	A1381	T_A = 25°C	6.00	–	9.00	mV/G
		A1382		4.00	–	6.25	mV/G
		A1383		2.75	–	4.25	mV/G
		A1384		2.00	–	3.00	mV/G
Sensitivity Programming Bits				–	6	–	bit
Average Sensitivity Step Size[8,9]	$Step_{SENS}$	A1381	T_A = 25°C	90	110	130	µV/G
		A1382		55	75	95	µV/G
		A1383		35	55	75	µV/G
		A1384		28	35	42	µV/G
Sensitivity Programming Resolution[10]	Err_{PGSENS}	T_A = 25°C		–	$Step_{SENS}$ × ±0.5	–	mV/G
SENSITIVITY TC PROGRAMMING							
Guaranteed Sensitivity Temperature Coefficient Range[6]	TC_{Sens}	T_A = 150°C		0.00	–	0.095	%/°C
Sensitivity Temperature Coefficient Programming Bits				–	3	–	bit
Average Sensitivity Temperature Coefficient Step Size[6]	$Step_{TCSENS}$	T_A = 150°C		–	0.03	–	%/°C
Sensitivity Temperature Coefficient Programming Resolution[6]	$Err_{PGTCSENS}$	T_A = 150°C		–	$Step_{TCSENS}$ x ±0.5	–	%/°C
POLARITY PROGRAMMING							
Polarity Programming Bit[12]	POL			–	1	–	bit
LOCK BIT PROGRAMMING							
Overall Programming Lock Bit	LOCK			–	1	–	bit

OPERATING CHARACTERISTICS (continued), valid over full operating temperature range, T_A, C_{BYPASS} = 0.1 µF, V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ERROR COMPONENTS						
Linearity Sensitivity Error	Lin_{ERR}		–	±1.5	–	%
Symmetry Sensitivity Error	Sym_{ERR}		–	±1.5	–	%
Ratiometry Quiescent Voltage Output Error[13]	$Rat_{ERRVOUT(Q)}$		–	±1.5	–	%
Ratiometry Sensitivity Error[13]	$Rat_{ERRSens}$		–	±1.5	–	%
Ratiometry Clamp Error[14]	Rat_{ERRCLP}	T_A = 25°C	–	±1.5	–	%
DRIFT CHARACTERISTICS						
Quiescent Voltage Output Drift Through Temperature Range	$\Delta V_{OUT(Q)}$	A1381, T_A = 150°C	–	–	±60	mV
		A1382, T_A = 150°C	–	–	±50	mV
		A1383, T_A = 150°C	–	–	±40	mV
		A1384, T_A = 150°C	–	–	±40	mV
Sensitivity Drift Through Temperature Range[15]	$\Delta Sens_{TC}$		–	±3	–	%
Sensitivity Drift Due to Package Hysteresis[1]	$\Delta Sens_{PKG}$	T_A = 25°C; after temperature cycling	–	±2	–	%

[1]See Characteristic Definitions section
[2]f_C varies up to approximately ± 20% over the full operating ambient temperature range, T_A, and process.
[3]Unit of measure (phase degrees) in reference to the magnetic input signal
[4]Sens, $V_{OUT(Q)}$, $V_{CLP(LOW)}$, and $V_{CLP(HIGH)}$ scale with V_{CC} due to ratiometry.
[5]Raw device characteristic values before any programming
[6]Programmed at 150°C and calculated relative to 25°C
[7]$V_{OUT(Q)}$(max) is the value available with all programming fuses blown (maximum programming code set). The $V_{OUT(Q)}$ range is the total range from $V_{OUT(Q)init}$ up to and including $V_{OUT(Q)}$(max) See Characteristic Definitions section.
[8]Step size is larger than required, in order to provide for manufacturing spread. See Characteristic Definitions section.
[9]Non-ideal behavior in the programming DAC can cause the step size at each significant bit rollover code to be greater than twice the maximum specified value of $Step_{VOUT(Q)}$, $Step_{SENS}$, or $Step_{TCSENS}$.
[10]Overall programming value accuracy. See Characteristic Definitions section.
[11]Sens(max) is the value available with all programming fuses blown (maximum programming code set) Sens range is the total range from $Sens_{init}$ up to and including Sens(max). See Characteristic Definitions section.
[12]Default polarity is for V_{OUT} voltage to increase with a positive (south polarity) field applied to the branded face of the device.
[13]Percent change from actual value at V_{CC} = 5 V, for a given temperature, over the guaranteed supply voltage operating range.
[14]Percent change from actual value at V_{CC} = 5 V, T_A = 25°C, over the guaranteed supply voltage operating range.
[15]Sensitivity drift from expected value at T_A after programming TC_{SENS}. See Characteristic Definitions section

B14 MOTORSPORT COMPONENTS **BOSCH**

THROTTLE POSITION SENSORS

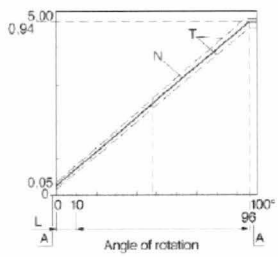
Purpose and Function.
Modern engine management systems require detailed information about throttle position and rate of change. As many vehicle systems are influenced by throttle activity including fuelling requirements, transmission control strategy and accessories such as air conditioning accurate data is essential, a simple switch cannot provide this detail.
The sensors described below are full range sensors capable of operating in clockwise or anti-clockwise directions and are compact for fitment in restricted spaces.
For more detailed information about these products refer to our website **www.bosch.com.au**

THROTTLE POSITION SENSOR TECHNICAL DATA

Part Number	Electrical Measurement Range	Operating Voltage	Connector	Drive Type	Direction of Rotation	Max. Circuit Current	Figure
0 280 122 001	< 86°	5.0	1 237 000 039	"D"	Optional	< 18 uA	A
0 261 211 003	< 93°	5.0	Non-Bosch	Dual "V"	C/Clockwise	< 10 mA	B
0 261 211 004	< 93°	5.0	Non-Bosch	Dual "V"	Clockwise	< 10 mA	B

Details of sensor 0 280 122 001 (fig. A)

Characteristic curve 1.
A Internal stop, L Positional tolerance of the wiper when fitted, N Nominal characteristic curve, T Tolerance limit.



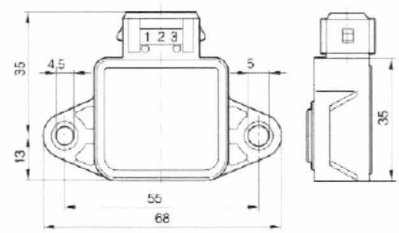


Image of sensor 0 261 211 003 / 4 (fig. B)

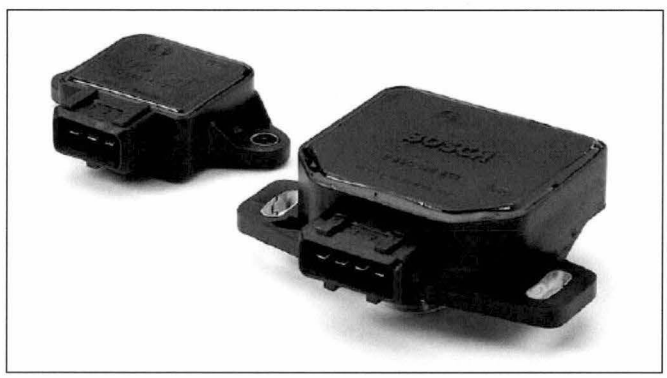
8 Angular-position sensors BOSCH

Throttle-valve angular-position sensor

Measurement of angles up to 88°

φ / R

- Potentiometic angular-position sensor with linear characteristic curve.
- Sturdy construction for extreme loading.
- Very compact.



Application
These sensors are used in automotive applications for measuring the angle of rotation of the throttle valve. Since these sensors are directly attached to the throttle-valve housing at the end of the throttle-shaft extension, they are subject to extremely hostile underhood operating conditions. To remain fully operational, they must be resistant to fuels, oils, saline fog, and industrial climate.

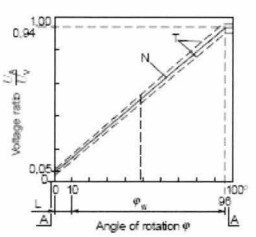
Design and function
The throttle-valve angular-position sensor is a potentiometric sensor with a linear characteristic curve. In electronic fuel injection (EFI) engines it generates a voltage ratio which is proportional to the throttle valve's angle of rotation. The sensor's rotor is attached to the throttle-valve shaft, and when the throttle valve moves, the sensor's special wipers move over their resistance tracks so that the throttle's angular position is transformed into a voltage ratio. The throttle-valve angular-position sensor's are not provided with return springs.

Design
The position sensor 0 280 122 001 has one linear characteristic curve.
The position sensor 0 280 122 201 has two linear characteristic curves.
This permits particularly good resolution in the angular range 0°...23°.

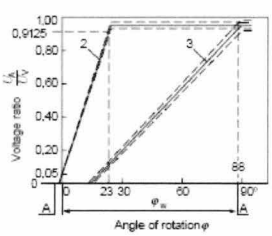
Explanation of symbols
U_A Output voltage
U_V Supply voltage
φ Angle of rotation
U_{A2} Output voltage, characteristic curve 2
U_{A3} Output voltage, characteristic curve 3

Accessories for 0 280 122 001	
Connector	**1 237 000 039**
Accessories for 0 280 122 201	
Plug housing	**1 284 485 118**
Receptacles, 5 per pack, Qty. required: 4	**1 284 477 121**
Protective cap, 5 per pack, Qty. required: 1	**1 280 703 023**

Characteristic curve 1.
A Internal stop, L Positional tolerance of the wiper when fitted, N Nominal characteristic curve, T Tolerance limit, φ_w Electrically usable angular range.



Characteristic curves 2 and 3.
A Internal stop,
φ_w Electrically usable angular range.



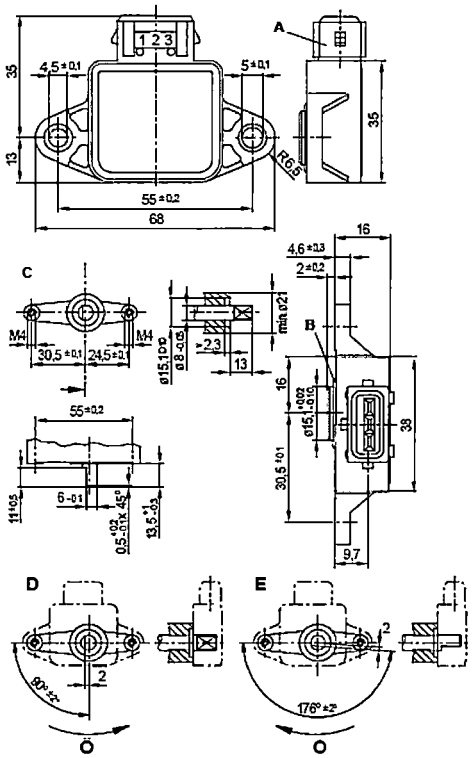
Technical data / Range

Part number		**0 280 122 001**	**0 280 122 201**
Diagram		1; 2	3
Useful electrical angular range	Degree	≤ 86	≤ 88
Useful mechanical angular range	Degree	≤ 86	≤ 92
Angle between the internal stops (must not be contacted when sensor installed)	Degree	≥ 95	–
Direction of rotation		Optional	Counterclockwise
Total resistance (Terms. 1–2)	kΩ	2 ±20 %	–
Wiper protective resistor (wiper in zero setting, Terms. 2–3)	Ω	710...1380	–
Operating voltage U_V	V	5	5
Electrical loading		Ohmic resistance	Ohmic resistance
Permissible wiper current	µA	≤ 18	≤ 20
Voltage ratio from stop to stop Chara. curve 1		$0.04 \le U_A/U_V \le 0.96$	–
Voltage ratio in area 0...88 °C Chara. curve 2		–	$0.05 \le U_{A2}/U_V \le 0.985$
Chara. curve 3		–	$0.05 \le U_{A3}/U_V \le 0.970$
Slope of the nominal characteristic curve	deg^{-1}	0.00927	–
Operating temperature	°C	–40...+130	–40...+85
Guide value for permissible vibration acceleration	$m \cdot s^{-2}$	≤ 700	≤ 300
Service life (operating cycles)	Mio	2	1.2

Dimension drawings.
A Plug-in connection,
B O-ring 14.65 x 2 mm,
C Fixing dimensions for throttle-valve housing, D Clockwise rotation [1]),
E Counterclockwise rotation [1]), Ö Direction of throttle-valve opening
[1]) Throttle valve in idle setting

F O-ring 16.5 x 2.5 mm, G 2 ribs, 2.5 mm thick,
H Plug-in connection, I Blade terminal,
K This mounting position is only permissible when the throttle-valve shaft is sealed against oil, gasoline, etc., Ö Direction of throttle-valve opening,
L Fixing dimensions for throttle-valve potentiometer.

0 280 122 001



0 280 122 201

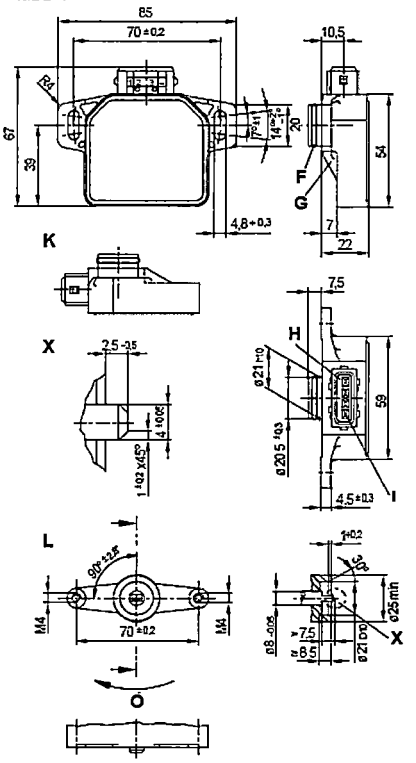


Diagram 1.

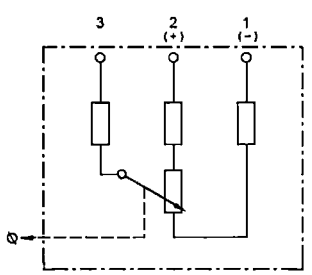


Diagram 2.

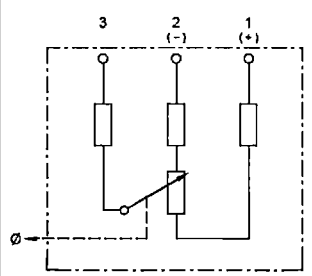
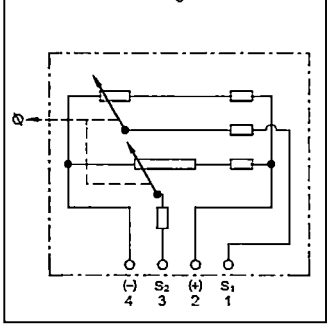


Diagram 3.
Throttle valve in idle setting



AutoXpert - Holden Calais 88-89 VN 3.8l V6 EFI Page 1

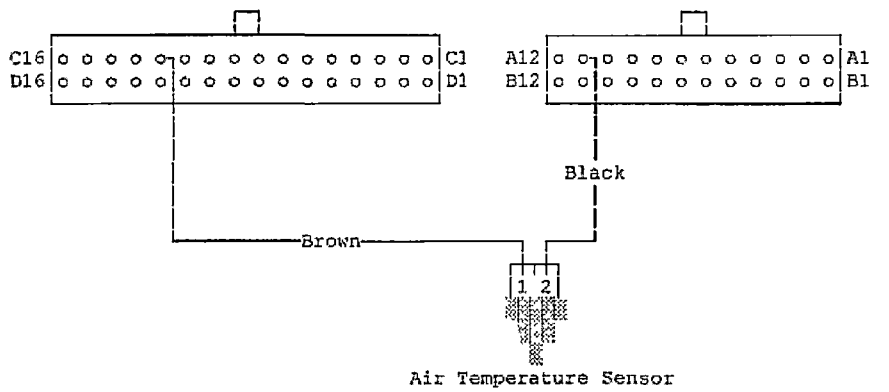
The MAT sensor is located in the inlet manifold, on passenger side of engine.

ECM C12 sends a 5.0 volt reference signal out to the MAT sensor. The sensor is a temperature variable resistor and is connected to earth through the ECM A11. Voltage will vary between 1.0 and 5.0 volts, depending on air temperature.

ECM A11 supplies the ground circuit for the MAT sensor, and is connected through the ECM circuitry to engine ground. Voltage on this line should always be zero volts. Voltage appearing on this terminal indicates:

a. Faulty connector on the ECM.
b. Problem with one of the other main ECM grounds.
c. Faulty ECM.

View looking into ECM connector (wire side).



TESTING
~~~~~~~
ECM C12. MAT Sensor Input
~~~~~~~~~~~~~~~~~~~~~~~~~
Ignition ON. Engine OFF.

Voltage should be 1.0 to 2.0 volts.

Engine RUNNING.
Voltage will be slightly higher - 2.0 to 3.0 volts.

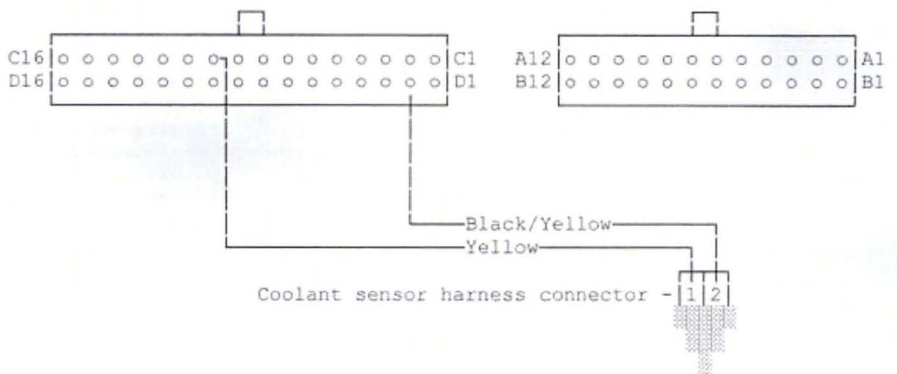
MAT Sensor Resistance
~~~~~~~~~~~~~~~~~~~~~
Ignition OFF.

Air temperature	Resistance
20° C	3,400 Ω
38° C	1,800 Ω
70° C	450 Ω
100° C	185 Ω
~~~~~~~~~~~~~~~~~~~~~

AutoXpert - Holden Calais 88-89 VN 3.81 V6 EFI Page 1

The ECT sensor is mounted in the coolant stream. It is found at the rear of the engine, adjacent to the thermostat housing.

View looking into ECM connector (wire side).



ECM C10 sends a 5.0 volt reference signal to the ECT sensor. The sensor is a temperature variable resistor, connected to earth through ECM D2.

Voltage on C10 will vary between 4.0 to 4.5 volts with a COLD engine and should DECREASE as engine temperature INCREASES. This voltage, at normal operating temperature, should be between 1.5 and 2.0 volts.

ECM D2 is connected to engine earth via the internal circuitry of the ECM. Voltage on this line should always be zero volts. Voltage appearing here indicates either a faulty connector on the ECM, or a faulty ECM.

A unique feature to the Delco system is that the ECM internally changes the resistance of the ECT circuit using an internal resistor network to increase the resolution of the signal. This is done in 2 stages. The first stage is when the engine temperature is below 51°C while the second stage is when the engine temperature is above 51°C.

Note: Remember, the ECM internally switches the amount of resistance on the circuit and not the voltage. So when testing the reference signal to the ECT with the harness disconnected it will always be 5 volts.

** NOTE ** If battery has been disconnected, start engine and allow it to IDLE for a few moments before starting voltage tests.

POSSIBLE FAULTS

ECT voltage too HIGH

Possible faulty earth on ECM D2.
Possible open circuit in ECT harness.
Possible open circuit in sensor or faulty sensor connector.
Thermostat stuck open or not installed.

ECT voltage too LOW

ECM C10 may be grounded.
ECM reference output voltage may be too low

Disconnect the CTS and recheck voltage.
If voltage is under 4.5 volts, remember that if the wire is pinched between the connector and the ECM, it will read low voltage. If voltage is low, make sure you check voltage at the ECM with the wire disconnected.

CHECKING SENSOR RESISTANCE

Sensor resistance values should be in the range shown below:

temperature	Resistance
20° C	2,500 Ω
30° C	1,800 Ω
40° C	1,200 Ω
70° C	450 Ω
90° C	250 Ω
100° C	190 Ω
110° C	110 Ω

** NOTE ** In the event of an ECT sensor failure (temperature unknown), the ECM should turn ON the engine cooling fan.
If engine fan will not turn OFF, ECT sensor is always suspect.

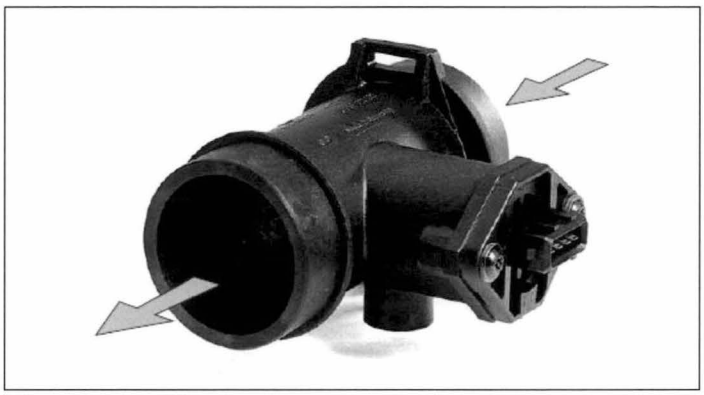
54 Air-mass meters BOSCH

Hot-film air-mass meter, type HFM 2

Measurement of air-mass throughflow up to 1080 kg / h

Q_m / U

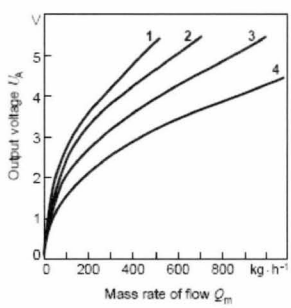
- Measurement of air mass (gas mass) throughflow per unit of time, independent of density and temperature.
- Extensive measuring range.
- Highly sensitive, particularly for small changes in flow rate.
- Wear-free since there are no moving parts.
- Insensitive to dirt and contamination.



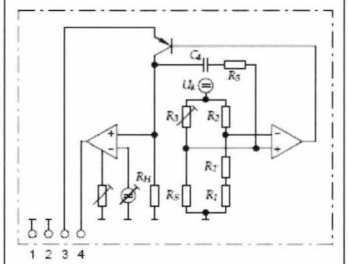
Application
Measurement of air-mass flow rate to provide data needed for clean combustion. Air-mass meters are suitable for use with other gaseous mediums.

Design and function
The sensor element comprises a ceramic substrate containing the following thick-film resistors which have been applied using silk-screen printing techniques: Air-temperature-sensor resistor R_θ, heater resistor R_H, sensor resistor R_S, and trimmer resistor R_1.
The heater resistor R_H maintains the platinum metallic-film resistor R_S at a constant temperature above that of the incoming air. The two resistors are in close thermal contact.
The temperature of the incoming air influences the resistor R_θ with which the trimmer resistor R_1 is connected in series. Throughout the complete operating-temperature range it compensates for the bridge circuit's temperature sensitivity. Together with R_2 and R_θ, R_1 forms one arm of the bridge circuit, while the auxiliary resistor R_3 and sensor resistor R_S form the other arm. The difference in voltage between the two arms is tapped off at the bridge diagonal and used as the measurement signal.
The evaluation circuit is contained on a second thick-film substrate. Both hybrids are integrated in the plastic housing of the plug-in sensor.
The hot-film air-mass meter is a thermal flowmeter. The film resistors on the ceramic substrate are exposed to the air mass under measurement. For reasons associated with flow, this sensor is far less sensitive to contamination than, for example, a hot-wire air-mass meter, and there is no need for the ECU to incorporate a self-cleaning burn-off function.

Characteristic curves.



Operating principle.

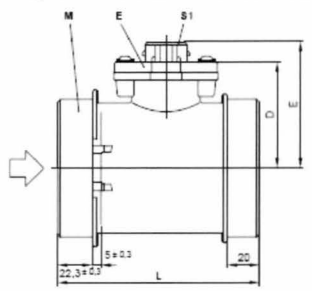


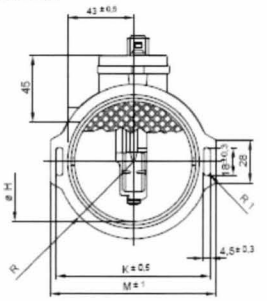
Technical data / Range

Part number		0 280 217 102	0 280 217 120 0 280 217 107	0 280 217 519	0 280 217 801
Characteristic curve		1	2	3	4
Installation length L	mm	130	130 96	130	130
Air-flow measuring range	kg · h⁻¹	10...350	10...480	12...640	20...1080
Accuracy referred to measured value	%	±4	±4	±4	±4
Supply voltage	V	14	14	14	14
Input current					
at 0 kg · h⁻¹	A	≤ 0,25	≤ 0,25	≤ 0,25	≤ 0,25
at $Q_{m\,nom.}$	A	≤ 0,8	≤ 0,8	≤ 0,8	≤ 0,8
Time constant [1]	ms	≤20	≤20	≤20	≤20
Temperature range					
Sustained	°C	–30...+110	–30...+110	–30...+110	–30...+110
Short-term	°C	–40...+125	–40...+125	–40...+125	–40...+125
Pressure drop at nominal air mass hPa	mbar	<15	<15	<15	<15
Vibration acceleration max.	m · s⁻²	150	150	150	150

[1] In case of sudden increase of the air-mass flow from 10 kg · h⁻¹ auf 0.7 $Q_{m\,nominal}$, time required to reach 63% of the final value of the air-mass signal.

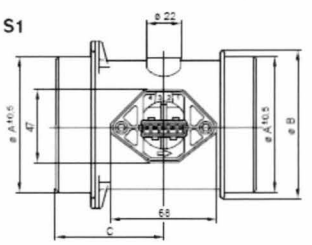
Dimension drawings.
E Plug-in sensor, **M** Measurement venturi, S1/S2 Plug connection



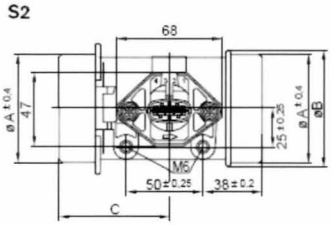


ØA	ØB	C	D	E	H	K	L	M	R	Measure-ment venturi	Plug-in connection	Part number
60	66	70	73	86	33	75	130	82	37	KS	S1	**0 280 217 102**
70	76	50	69	82	34.8	–	96	–	42	KS	S1	**0 280 217 107**
70	76	70	69	82	33.5	85	130	92	42	KS	S2	**0 280 217 120**
80	86	70	73	86	39	–	130	–	–	KS	S2	**0 280 217 519**
95.6	102	70	76.2	91.2	45	110	130	117	54	Alu	S1	**0 280 217 801**

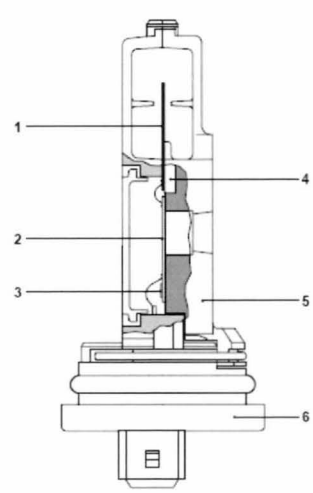
S1



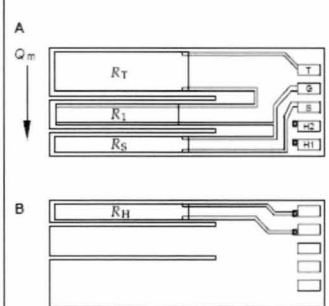
S2



Plug-in sensor.
1 Sensor, **2** Hybrid, **3** Power module, **4** Mounting plate, **5** Heat sink, **6** Plug housing



Sensor element with thick-film resistors.
Q_M Mass rate of flow, R_1 Trimmer resistor, R_H Heater resistor, R_S Sensor resistor, R_T Air-temperature measuring resistor, **A** Front, **B** Rear



Installation instructions
Water and other liquids must not collect in the measurement venturi. The measurement venturi must therefore be inclined by at least 5° relative to the horizontal. Since care must be taken that the intake air is free of dust, it is imperative that an air filter is fitted.

Explanation of symbols:
R_1 Trimmer resistor
R_2, R_3 Auxiliary resistors
R_5, C_4 RC element
R_H Heater resistor
R_S Platinum metal-film resistor
R_T Resistance of the air-temperature-sensor resistor
U_K Bridge supply voltage
U_A Output voltage
U_V Supply voltage

Connector-pin assignment
Pin 1 Ground
Pin 2 U_A (–)
Pin 3 U_V
Pin 4 U_A (+)

Accessories

For 0 280 217 102, .. 107, .. 801	
Plug housing	**1 284 485 118**
Receptacle	**1 284 477 121** [1]
Protective cap	**1 280 703 023** [1]

Each 4-pole plug requires 1 plug housing, 4 receptacles, and 1 protective cap.
[1] Quantity 5 per package

For 0 280 217 120, .. 519

Desig-nation	For conductor cross-section	Part number
Plug housing	–	**1 928 403 112**
Contact pin	0.5...1.0 mm²	**1 987 280 103**
	1.5...2.5 mm²	**1 987 280 105**
Individual gasket	0.5...1.0 mm²	**1 987 280 106**
	1.5...2.5 mm²	**1 987 280 107**

Each 4-pole plug requires 1 plug housing, 4 contact pins, and 4 individual gaskets.

Note
For automotive applications, original AMP crimping tools must be used.

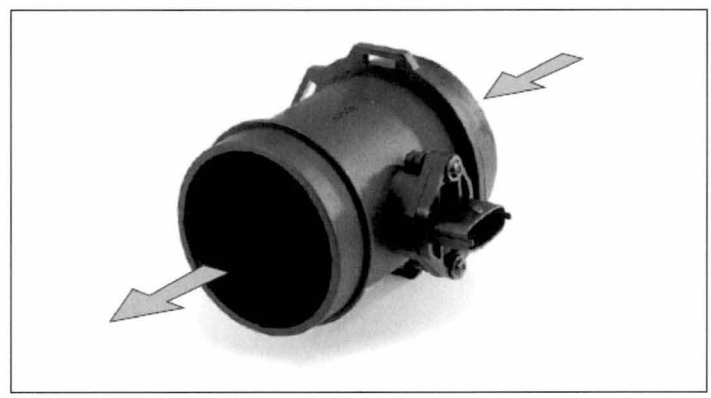
56 Air-mass meters BOSCH

Hot-film air-mass meter, Type HFM 5

Measurement of air-mass throughflow up to 1000 kg/h

Q_m/U

- Compact design.
- Low weight.
- Rapid response.
- Low power input.
- Return-flow detection.



Application

In order to comply with the vehicle emission limits demanded by law, it is necessary to maintain a given air/fuel ratio exactly.
This requires sensors which precisely register the actual air-mass flow and output a corresponding electrical signal to the open and closed-loop control electronics.

Design

The micromechanical sensor element is located in the plug-in sensor's flow passage. This plug-in sensor is suitable for incorporating in the air filter or, using a measurement venturi, in the air-intake passages. There are different sizes of measurement venturi available depending upon the air throughflow. The micromechanical measuring system uses a hybrid circuit, and by evaluating the measuring data is able to detect when return flow takes place during air-flow pulsation.

Operating principle

The heated sensor element in the air-mass meter dissipates heat to the incoming air. The higher the air flow, the more heat is dissipated. The resulting temperature differential is a measure for the air mass flowing past the sensor.
An electronic hybrid circuit evaluates this measuring data so that the air-flow quantity can be measured precisely, and its direction of flow.
Only part of the air-mass flow is registered by the sensor element. The total air mass flowing through the measuring tube is determined by means of calibration, known as the characteristic-curve definition.

Technical data / range

Nominal supply voltage U_N	14 V
Supply-voltage range U_V	8...17 V
Output voltage U_A	0...5 V
Input current I_V	< 0.1 A
Permissible vibration acceleration	≤ 150 ms^{-2}
Time constant τ_{63} [1]	≤ 15 ms
Time constant τ_Δ [2]	≤ 30 ms
Temperature range	–40...+120 °C [3]

Part number	0 280 217 123	0 280 218 019	0 280 217 531	0 280 218 008	0 281 002 421
Measuring range Q_m	8...370 kg/h	10...480 kg/h	12...640 kg/h	12...850 kg/h	15...1000 kg/h
Accuracy [4]	≤ 3%	≤ 3%	≤ 3%	≤ 3%	≤ 3%
Fitting length L_E	22 mm	22 mm	22 mm	16 mm	22 mm
Fitting length L_A	20 mm	20 mm	20 mm	16 mm	20 mm
Installation length L	96 mm	96 mm	130 mm	100 mm	130 mm
Connection diam. D	60 mm	70 mm	80 mm	86/84 mm [6]	92 mm
Venturi ID	50 mm	62 mm	71 mm	78 mm	82 mm
Pressure drop at nominal air mass [5]	< 20 hPa	< 15 hPa	< 15 hPa	< 15 hPa	< 15 hPa
Temperature sensor	Yes	Yes	Yes	No	Yes
Version	1	2	3	4	5

[1] In case of sudden increase of the air-mass flow from 10 kg · h^{-1} auf 0,7 $Q_{m\ nominal}$, time required to reach 63% of the final value of the air-mass signal.
[2] Period of time in case of a throughflow jump of the air mass $|\Delta m/m| \leq 5\%$.
[3] For a short period up to +130 °C.
[4] $|\Delta Q_m/Q_m|$: The measurement deviation ΔQ_m from the exact value, referred to the measured value Q_m.
[5] Measured between input and output
[6] Inflow/outflow end

Accessories for connector

Plug housing	Contact pins	Individual gaskets	For conductor cross-section
1 928 403 836	**1 987 280 103**	**1 987 280 106**	0.5...1 mm²
	1 987 280 105	**1 987 280 107**	1.5...2.5 mm²

Note: Each 5-pole plug requires 1 plug housing, 5 contact pins, and 5 individual gaskets.
For automotive applications, original AMP crimping tools must be used.

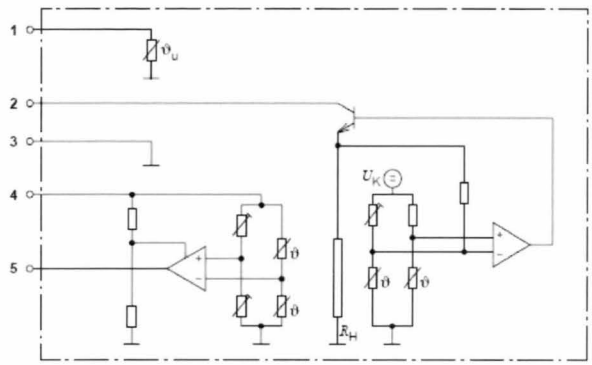
Application

In internal-combustion engines, this sensor is used for measuring the air-mass flow so that the injected fuel quantity can be adapted to the presently required power, to the air pressure, and to the air temperature.

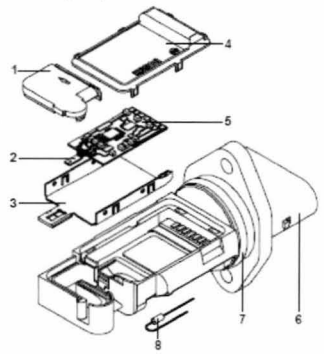
Explanation of symbols

Q_m	Air-mass flow rate
ΔQ_m	Absolute accuracy
$\Delta Q_m/Q_m$	Relative accuracy
τ_Δ	Time until measuring error is ≤ 5%
τ_{63}	Time until measured-value change 63%

Function diagram with connector-pin assignment.
1 Additional temperature sensor ϑ_U (not on version 4, Part number 0 280 218 008),
2 Supply voltage U_V, 3 Signal ground, 4 Reference voltage 5 V, 5 Measurement signal U_A.
ϑ Temperature-dependence of the resistor, R_H Heater resistor, U_K Constant voltage



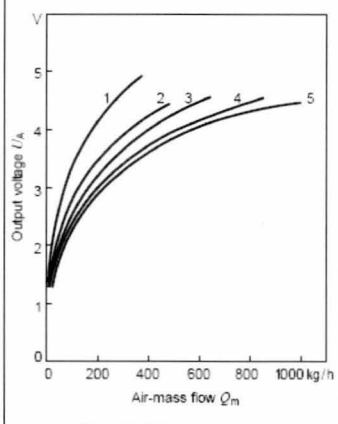
HFM 5 plug-in sensor design.
1 Measuring-passage cover, 2 Sensor,
3 Mounting plate, 4 Hybrid-circuit cover,
5 Hybrid, 6 Plug-in sensor, 7 O-ring,
8 Auxiliary temperature sensor.



Output voltage U_A = f(Q_m) of the air-mass meter

Part number	0 280 217 123	0 280 218 019	0 280 217 531	0 280 218 008	0 280 002 421
Characteristic curve	1	2	3	4	5
Q_m/kg/h	U_A/V	U_A/V	U_A/V	U_A/V	U_A/V
8	1.4837	1.2390	–	–	–
10	1.5819	1.3644	1.2695	–	–
15	1.7898	1.5241	1.4060	1.3395	1.2315
30	2.2739	1.8748	1.7100	1.6251	1.4758
60	2.8868	2.3710	2.1563	2.0109	1.8310
120	3.6255	2.9998	2.7522	2.5564	2.3074
250	4.4727	3.7494	3.5070	3.2655	2.9212
370	4.9406	4.1695	3.9393	3.6717	3.2874
480	–	4.4578	4.2349	3.9490	3.5461
640	–	–	4.5669	4.2600	3.8432
850	–	–	–	4.5727	4.1499
1000	–	–	–	–	4.3312

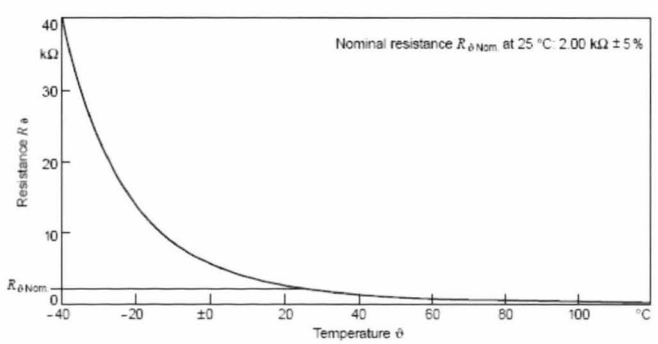
Air-mass meter output voltage.



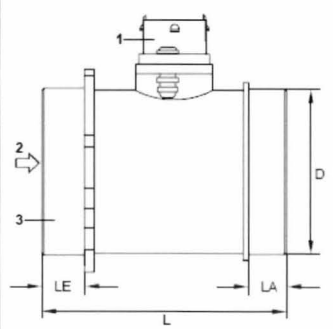
Temperature-dependence R_ϑ = f(ϑ) of the temperature sensor

Temperature ϑ	°C	–40	–30	–20	–10	±0	10	20	30	40
Resistance R_ϑ	kΩ	39.26	22.96	13.85	8.609	5.499	3.604	2.420	1.662	1.166
Temperature ϑ	°C	50	60	70	80	90	100	110	120	130
Resistance R_ϑ	Ω	835	609	452	340	261	202	159	127	102

Temperature-resistance diagram of the temperature sensor.



Dimensions overview of the HFM 5.
1 Plug-in sensor, 2 Throughflow direction,
3 Measurement venturi.



APPENDIX C
MOTEC ECU SPECIFICATION

General

- Microprocessor 32 Bit 33 MHz with Time CO Processor
- Quality Standard ISO 9001
- Manufacturing Standard IPC-S-815-A Class 3 High Reliability
- Warranty 1 year Parts & Labour
- Burn In -50 to 70°C for 32 Hrs
- ECU Control Software stored in updatable FLASH memory
- High RFI Immunity
- Low heat generation
- Battery transient protection
- Environmentally sealed electronics
- Water proof connector with gold plated contacts
- Case Size 120 x 100 x 36 mm (4.7 x 3.9 x 1.4 inches)
- Weight 0.4 kg (14 oz)
- Cylinders 1,2,3,4,6,8,12
- Engines 2 stroke, 4 stroke, Rotary (1~4), Odd or Even fire
- Maximum RPM >15,000 RPM

Fuel Calibration

- Accuracy 0.00001 seconds
- All RPM & Load sites are user programmable
- Main Table (3D) 40 RPM sites x 21 Load sites (840 points)
- End of Injection (3D) 20 RPM sites x 6 Load sites
- Overall Trim ±99 %
- Individual Cylinder Trim ±99 %
- Individual Cylinder Tables (3D) 20 RPM sites x 11 Load sites
- Hi / Lo Injector Balance (3D) 20 RPM sites x 11 Load sites
- Hi / Lo End of Injection (3D) 20 RPM sites x 11 Load sites
- Eng Temp & Air Temp comps
- MAP modifier compensation
- Two Auxiliary compensation
- Injector Dead Time Compensation
- Accel Clamp, Decay & Sensitivity
- Deccel Clamp, Decay & Sensitivity
- Cold Start

Injection

- 4 group sequential
- User programmable injector current 0.5~12 Amps peak
- Battery Comp to suit any injector

Boost Control Calibration

- Main Table (3D) 20 RPM sites x 10 Throttle sites or 10 Gear sites
- Overall Trim
- Engine Temp & Air Temp comps
- One Auxiliary compensation

Ignition Outputs

- Up to 4 Ignition Outputs
- One output may drive up to 8 coils using the MoTeC Ignition Expander
- Versatile Ignition Interface allows connection to most OEM ignition systems including:
    - Nissan Multi Coil modules
    - GM EST DFI systems
    - FORD EDIS DFI systems
    - Mazda Rotary DFI modules
    - Many Others

Ignition Calibration

- Accuracy 0.25 degrees
- All RPM & Load sites are user programmable
- Main Table (3D) 40 RPM sites x 21 Load sites (840 points)
- Overall Trim ±99 %
- Individual Cylinder Trim ±99 %
- Individual Cylinder Tables (3D) 20 RPM sites x 11 Load sites
- Rotary Split 20 RPM x 11 Load
- Eng Temp & Air Temp comps
- MAP compensation
- Two Auxiliary compensations
- Dwell Time 20 RPM x11 Battery
- Odd fire engine capability Each Top Dead Centre angle may be specified Resolution 0.5 degree

Trigger Sensors

- Directly compatible with most OEM trigger systems including :
  - HALL, Magnetic and Optical types
  - Multi Tooth (eg Mazda and Toyota)
  - 1 or 2 Missing Teeth (eg Porsche)
  - Many other special types, eg. Ford Narrow Tooth, Nissan Optical, Harley Davidson

Data Logging

- Optional Logging memory allows logging of all ECU parameters
- Memory Size 512 KByte
- Logging Rate 1~20 sets / sec
- Logging Time 38 minutes at 5 sets / sec (28 Parameters+Diags) PC Software is available for analysis of the logged data.

Air Fuel Ratio Sensor

- High accuracy Wide Band Air Fuel Ratio Sensor Input (Optional)
- Range 0.75 to 1.20 Lambda
- Resolution 0.01 Lambda

Other Sensors

- Throttle Pos, Manifold Pressure, Engine Temp & Air Temp
- 2 Auxiliary Sensor inputs
- 2 Digital / Speed Inputs

Special Functions

- Traction Control & Launch Control (2 wheel speed sensors) (or 4 sensors using the MoTeC TC Mux)
- Gear Change Ignition Cut
- Wide Band or Narrow Band Air Fuel Ratio Control (3D mapped)
- Over Run Boost Enhancement
- Warning Alarms (Sensor HI / LO)
- Gear Detection Ground Speed Limiting
- Dual RPM Limit
- Nitrous Oxide Enrich / Retard
- Air Conditioner Request

- Over Run Fuel Cut
- Sensor Calibration Tables RPM Limit Hard or Soft Cut Fuel and / or Ignition Cut

Auxiliary Outputs

- Four general purpose outputs (3 shared with ignition outputs)
- The outputs may be used for :
    - Turbo Wastegate Control
    - Idle Speed Control
    - Fuel Used Pulse
    - Tacho Output
    - Shift Light (Gear Dependent)
    - Driver Warning Alarm
    - RPM / Load dependant device
    - User Defined Table (20x11) with definable axis parameters
    - Slip Warning
    - Fuel Pump Relay
    - Thermatic Fan
    - Air Conditioner Fan or Clutch

Diagnostics

- Injectors Open Circuit, Short Circuit, Peak Current Not Reached
- Sensors Open & Short circuit
- Operating Errors RPM Limit Exceeded, Injector Overduty, Over Boost, Low Battery, REF Error etc.

Operating Conditions

- Internal Temp Range -10~85°C
- Ambient Temp -10~70°C (Depending on load & ventilation)
- Operating Voltage 6~22V DC
- Operating Current 0.4 A max
- Reverse Battery External Fuse

Calibration

- Personal Computer (see PC Requirements)

Telemetry Link

- Optional Telemetry Link allows real time monitoring and data logging

APPENDIX D
MOTOROLA 68HC12 MICRO-CONTROLLER SPECIFICATION

Features include:

- 16-bit CPU12:
  - Upwardly compatible with the M68HC11 instruction set
  - Interrupt stacking and programmer's model identical to the M68HC11
  - 20-bit arithmetic logic unit (ALU)
  - Instruction queue
  - Enhanced indexed addressing
  - Fuzzy logic instructions

- Multiplexed bus:
  - Single chip or expanded
  - 16-bit by 16-bit wide or 16-bit by 8-bit narrow modes

- Memory:
  - 32-Kbyte FLASH electrically erasable, programmable read-only memory (EEPROM) with 2-Kbyte erase-protected boot block — MC68HC912B32 and MC68HC912BC32 only
  - 32-Kbyte ROM — MC68HC12BE32 and MC68HC12BC32 only
  - 768-byte EEPROM
  - 1-Kbyte random-access memory (RAM) with single-cycle access for aligned or misaligned read/write

- 8-channel, 10-bit analog-to-digital converter (ATD)

- 8-channel standard timer module (TIM) — MC68HC912B32 and MC68HC(9)12BC32 only:
  - Each channel fully configurable as either input capture or output compare
  - Simple pulse-width modulator (PWM) mode
  - Modulus reset of timer counter

APPENDIX E
EMBEDDED SOURCE CODES FOR HC12 BOARD

HC12 main code "cinj5.c"

```
#include <912b32.h>
#pragma interrupt_handler TC4_Handler
#pragma interrupt_handler TC0_Handler

//Injection variables
#define inj_start           *(unsigned char  *) 0x0800
#define inj_dura            *(unsigned char  *) 0x0801

#define l_inj_start_tick    *(unsigned long  *) 0x0802
#define temp_inj_start_tick *(unsigned short *) 0x0804//2LSB of the long
#define inj_start_tick      *(unsigned short *) 0x0806

#define l_inj_dura_tick     *(unsigned long  *) 0x0808
#define temp_inj_dura_tick  *(unsigned short *) 0x080A//2LSB of the long
#define inj_dura_tick       *(unsigned short *) 0x080C

#define inj_diff_tick       *(unsigned short *) 0x080E
#define inj_stop_tick       *(unsigned short *) 0x0810
#define inj_instance        *(unsigned char  *) 0x0812
#define inj_event           *(unsigned short *) 0x0813

//Common variables
#define new_event           *(unsigned short *) 0x0815
#define old_event           *(unsigned short *) 0x0817
#define delta               *(unsigned long  *) 0x0819
#define resolution          *(unsigned long  *) 0x081D
#define temp                *(unsigned short *) 0x0821
#define lag_tick            *(unsigned short *) 0x0823
#define data_mux            *(unsigned char  *) 0x0825
#define engine_status       *(unsigned char  *) 0x0826
#define int_enable_status   *(unsigned char  *) 0x0827

void TC4_Handler(void)  //Input capture 4 - CAM signal
{
  new_event=TCNT;
  PORTP=0x00;
  if (TFLG2&0x80==0x80) //Check TOF counter overflow flag
  {
      temp=0xFF-old_event;
      delta=temp+new_event;
      TFLG2=0x80;     //Clear TOF flag to allow next interrupt
  }

  else
  {
    delta=new_event-old_event;
  }
  inj_instance = 0;       // Reset injection instance variable
  TC0 = new_event+inj_start_tick;
```

```
  TFLG1 = 0x11;           //Clear interrupt flag for TC0&TC4
  old_event=new_event;
}

//Timer TC0. instance monitoring interrupt routine
void TC0_Handler(void)
{
  inj_event=TCNT;
  switch (inj_instance)
  {
     case 0:TC0=inj_event+inj_dura_tick;//start of injection 0
            PORTP=PORTP|0b00000001;
             break;
      case 1:TC0=inj_event+inj_diff_tick; //end of injection 0
            PORTP=PORTP&0b11111110;
             break;
      case 2:TC0=inj_event+inj_dura_tick; //start of injection 1
             PORTP=PORTP|0b00000010;
             break;
      case 3:TC0=inj_event+inj_diff_tick; //end of injection 1
            PORTP=PORTP&0b11111101;
             break;
      case 4:TC0=inj_event+inj_dura_tick; //start of injection 2
             PORTP=PORTP|0b00000100;
             break;
      case 5:TC0=inj_event+inj_diff_tick; //end of injection 2
            PORTP=PORTP&0b11111011;
             break;
      case 6:TC0=inj_event+inj_dura_tick; //start of injection 3
            PORTP=PORTP|0b00001000;
             break;
     case 7:PORTP = 0x00;
  break;
default:break;

  }
  TFLG1=0x01;
  inj_instance+=1;
}
void main(void)
{
  INIT:
  asm("SEI");   //Disable interrupts
  HPRIO  = 0xE6;//Input capture TC4 has highest interrupt
                priority
  COPCTL = 0x00;                  //Disable Watchdog timer
  TIOS   = 0x03;//Output Compare 0,1-Input capture 4  Select
  TSCR  |= 0x80;                  //Timer Enable
  TMSK2 |= 0x05; //Prescaler Select = 32  (262.144ms<->65536
clock ticks
  TMSK1  = 0x11;                  //Interrupt on TC0&TC4 Enable
  TFLG1  = 0x11;              //Clear Interrupt flag for TC0&TC4
  TCTL2  = 0x00;    //Timer disconnected from output pin logic
  TC0    = TCNT+100;
  TCTL3  = 0x02;               //Capture on Falling Edge of TC4
  DDRP   = 0xFF;                  //PORTP pins are outputs
  PORTP  = 0x00;
```

```
inj_start = 0;
inj_dura  = 0;

inj_start_tick=0;
inj_dura_tick=0;
inj_diff_tick=0;
inj_stop_tick=0;

inj_instance = 0;
l_inj_start_tick=0;
l_inj_dura_tick=0;

DDRB=0x08;
DDRA=0x00;
DDRDLC=0X00;

PORTB=0x00;

data_mux=0x00;
engine_status=0;
int_enable_status = 0;

while (1)
{

  MAIN: engine_status=PORTDLC&0b00000001;

            if (engine_status==0)      //Engine stops
            {
              if (int_enable_status==1)
                {
                  asm("SEI");
                  int_enable_status=0;
                }
                goto MAIN;
            }

            if (engine_status==1)      //Engine running
            {
              if (int_enable_status==0)
                {
                  asm("CLI");
                  int_enable_status=1;
                }

              goto STROBE;
            }

STROBE: if ((PORTB&0x01)!=0x01)          //Check if STROBE=0
          {
            data_mux=PORTB&0b00000110;
            if (data_mux==0b00000100)
            {
              inj_start=PORTA;
            }
            if (data_mux==0b00000010)
```

```
            {
              inj_dura=PORTA;
                PORTB|=0x08;                          //Set ACK
              while ((PORTB&0x01)!=0x01)//Check if STROBE
                                        still=0
              {
              }
              PORTB&=0xF7;                        //Clear ACK
                goto CALC;
            }
            PORTB|=0x08;                              //Set ACK
            while ((PORTB&0x01)!=0x01)//Check if STROBE
                                    still=0
           {
            }
            PORTB&=0xF7;                        //Clear ACK
              goto STROBE;
          }

        CALC:
    resolution=(delta*10)/72;//Resolution: Clock
                             ticks/100deg
        lag_tick=delta/4;     //Lag angle 180 deg in ticks
        l_inj_start_tick=(inj_start*resolution)/100;
          l_inj_dura_tick=(inj_dura*resolution)/100;
          inj_start_tick=temp_inj_start_tick;
          inj_dura_tick=temp_inj_dura_tick;
          inj_diff_tick = lag_tick-inj_dura_tick;
        inj_stop_tick = inj_diff_tick-inj_start_tick;
  }
}
```

HC12 vector code "912b32.h"

```
#if defined(_HC12)
#pragma nonpaged_function _start
#endif
extern void _start(void);     /* entry point in crt.s */

extern void TC0_Handler(void);
extern void TC1_Handler(void);
extern void TC2_Handler(void);

extern void TC4_Handler(void);

#define DUMMY_ENTRY     (void (*)(void))0xFFFF

#pragma abs_address:0xffd0

void (*interrupt_vectors[])(void) =
      {
      DUMMY_ENTRY,       /* BDLC */              /* Key Wakeup J */
      DUMMY_ENTRY,       /* ATD */               /* ATD */
      DUMMY_ENTRY,       /* RESERVED */          /* SCI 1 */
      DUMMY_ENTRY,       /* SCI */
      DUMMY_ENTRY,       /* SPI */
      DUMMY_ENTRY,       /* PAIE */
      DUMMY_ENTRY,       /* PAO */
      DUMMY_ENTRY,       /* TOF */
      DUMMY_ENTRY,       /* HC12 TC7 */
      DUMMY_ENTRY,       /* TC6 */
      DUMMY_ENTRY,       /* TC5 */
     TC4_Handler, /* TC4 */
     DUMMY_ENTRY, /* TC3 */
     DUMMY_ENTRY, /* TC2 */
     DUMMY_ENTRY, /* TC1 */
      TC0_Handler,       /* TC0 */
      DUMMY_ENTRY,       /* RTI */
      DUMMY_ENTRY,       /* IRQ */
      DUMMY_ENTRY,       /* XIRQ */
      DUMMY_ENTRY,       /* SWI */
      DUMMY_ENTRY,       /* ILLOP */
      DUMMY_ENTRY,       /* COP */
      DUMMY_ENTRY,       /* CLM */
      _start       /* RESET */
      };
#pragma end_abs_address
```

APPENDIX F
EMBEDDED SOURCE CODES FOR PC104 BOARD

Header file "inj.h"

```
#include <conio.h>

#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream.h>
#include <string.h>
#include <time.h>

#define nspeed 8
#define nthrottle 11
#define H2filename "Hmaps.dat"
#define Petrolfilename "Pmaps.dat"

#define UP 'W'
#define up 'w'
#define DOWN 'S'
#define down 's'
#define LEFT 'A'
#define left 'a'
#define RIGHT 'D'
#define right 'd'

/*
+PARALLEL PORT (used for engine control data transfer to the HC12 board)
   CONTROL bit0=0  --> STROBE=1      // Data not available
   CONTROL bit0=1  --> STROBE=0      // Data available
   CONTROL bit1=0  --> AUTO_FEED= 1  //
   CONTROL bit1=1  --> AUTO_FEED= 0  //
   CONTROL bit2=0  --> INIT=0
   CONTROL bit2=1  --> INIT=1
   STATUS  bit3=x  --> ERROR=x
   STATUS  bit6=x  --> ACKNOWLEDGE=x
   STROBE - pin 1
   INIT - pin 16
   ACKNOWLEDGE - pin 10
   AUTO_FEED - pin 14
   ERROR - pin 15

+I/O PORT (PORT A ---> A3, A2, A1, A0 are used to indicate engine status:
   Port number = 0   --> Port A
   Port number = 1   --> Port B
   Port number = 2   --> Port C
                     _____________________________________________________
   Config_byte =   | DIOCTR | X | X | DIRA | DIRCH | X | DIRB | DIRCL |
                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   DIOCTR = 1  --> pin direction of C is controlled by DIRCH
```

```
   DIRA,DIRB,DIRCH,DIRCL = 0 ---> output, 1 ---> input
*/
#define DATA 0x0378
#define STATUS DATA+1
#define CONTROL DATA+2

enum bool{true,false};
//--------INJ2.CPP-------------------------------------------------------------
void MonitorSetup(void);
int KeyReady(void);
char GetKeyNumber(void);
void WriteStringXY(unsigned char x, unsigned char y, char *st);
void WriteIntXY(unsigned char x, unsigned char y, int number);
void WriteCharXY(unsigned char x,unsigned char y,char ch);
void ClearWindow(unsigned char x1,unsigned char y1,unsigned char x2,unsigned
char y2);
void ConvertDimension(unsigned char OneD[nthrottle*nspeed],unsigned char
TwoD[nthrottle][nspeed]);
bool WriteDataToFile(char *filename,unsigned char
map_start[nthrottle][nspeed],unsigned char map_duration[nthrottle][nspeed]);
bool ReadDataFromFile(char *filename,unsigned char
map_start[nthrottle][nspeed],unsigned char map_duration[nthrottle][nspeed]);
unsigned char CheckValueInRange(unsigned int value,unsigned int
*items,unsigned int n_items);
unsigned char ValueIndex(unsigned int value,unsigned int *items,unsigned int
n_items);
void SendInjectionData(unsigned int inj_start_param,unsigned int
inj_dura_param);
//--------INJ1.CPP-------------------------------------------------------------
int InitPROMBoard(void);
int ON_device(BYTE ID);
int OFF_device(BYTE ID);
int SetHC12BoardState(int state);  //state = HC12RUN or HC12STOP
void CAMInterruptHandler();
void SampleAnalogSignal(FLOAT &airmassvolt,FLOAT &airtempvolt,FLOAT
&engtempvolt,FLOAT &throttlevolt);
unsigned int EngineSpeedCalculation(void);
unsigned int ThrottlePositionCalculation(void);
void Initialization(void);
void Dispose(void);
void LoadDefaultH2FuelMaps(void);
void LoadH2FuelMapsFromFile(void);
void SaveH2FuelMapsToFile(void);
void LoadPetrolFuelMapsFromFile(void);
void SavePetrolFuelMapsToFile(void);
void MixingExperiment(void);
void StartInjectionTuning(void);
void DurationInjectionTuning(void);
void ViewStartInjectionMap(unsigned char x,unsigned char y,unsigned char
map_start[nthrottle][nspeed]);
void ViewDurationInjectionMap(unsigned char x,unsigned char y,unsigned char
map_duration[nthrottle][nspeed]);
void ViewStartInjectionH2Map(unsigned char x,unsigned char y);
void ViewDurationInjectionH2Map(unsigned char x,unsigned char y);
void ViewStartInjectionPetrolMap(unsigned char x,unsigned char y);
void ViewDurationInjectionPetrolMap(unsigned char x,unsigned char y);
void PrintMainMenu(void);
```

Main file "inj1.cpp"

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include "inj.h"
#include "dscud.h"

//--------------- Prediction part - declaration  -------------
#define LOWER_RPM 1100
#define UPPER_RPM 4202

#define LOWER_THROTTLE_V 0.445862
#define UPPER_THROTTLE_V 1.13

#define LOWER_AIRTEMP_V 2.5583
#define UPPER_AIRTEMP_V 4.06

#define LOWER_ENGTEMP_V 0.793762
#define UPPER_ENGTEMP_V 3.61

#define LOWER_AIRMASS_V 1.5802
#define UPPER_AIRMASS_V 2.754541


#define LOWER_H2_START 20
#define UPPER_H2_START 140

#define LOWER_H2_DURATION 20
#define UPPER_H2_DURATION 63

#define LOWER_PETROL_START 5
#define UPPER_PETROL_START 133

#define LOWER_PETROL_DURATION 42
#define UPPER_PETROL_DURATION 155


int nInputs=9;          //number of inputs in each model
int nOutputs=1;         //number of outputs in each model

int nHiddenNeuronsCO=5;   //number of hidden neurons in CO model
int nHiddenNeuronsHC=6;   //number of hidden neurons in HC model
int nHiddenNeuronsNO=6;   //number of hidden neurons in NO model

float *ptrInputs;          //Common for all predictive outputs
float *ptrHiddenOutputs;   //Common for all predictive outputs
```

```
float *ptrCOoutput;
float *ptrHCoutput;
float *ptrNOoutput;
char  *COweightfilename="COwts.out";
float *ptrCOinWeights;
float *ptrCOhiddenWeights;
char  *HCweightfilename="HCwts.out";
float *ptrHCinWeights;
float *ptrHChiddenWeights;
char  *NOweightfilename="NOwts.out";
float *ptrNOinWeights;
float *ptrNOhiddenWeights;

// ----------- Control part - declaration ----------------
#define ERROR_PREFIX "PROM Driver ERROR:"
#define BASE 0x0280

#define THROTTLE_VOLT_INTERVAL 4.0
#define THROTTLE_VOLT_MIN 1.0

#define DEVICE0 0 //Connected to B0
#define DEVICE1 1 //Connected to B1
#define DEVICE2 2 //Connected to B2
#define DEVICE3 3 //Connected to B3

#define HC12RUN 1
#define HC12STOP 0

BYTE result; // returned error code
DSCB dscb;   // handle used to refer to the board
DSCCB dsccb; // structure containing board settings
DSCADSETTINGS dscadsettings; // structure containing A/D conversion settings
DSCADSCAN dscadscan; // structure containing A/D scan settings
WORD* samples;  // sample readings
ERRPARAMS errorParams;  // structure for returning error code and error
string
DSCUSERINT dscuserint;
DSCAIOINT dscaioint;
DSCS dscs;
void (*pfunc);
unsigned char LSB=0,MSB=0;

BYTE config_byte;

FLOAT m_airmassvolt;
FLOAT m_airtempvolt;
FLOAT m_engtempvolt;
FLOAT m_throttlevolt;

unsigned int array_speed[nspeed] =
{1500,2000,2500,3000,3500,4000,4500,5000};

unsigned int array_throttle[nthrottle] = {0,10,20,30,40,50,60,70,80,90,100};

unsigned char default_H2H2fuelmap_start[nthrottle][nspeed] =
                              {{1,2,3,4,5,6,7,8},
                               {11,12,13,14,15,16,17,18},
```

```
                                {21,22,23,24,25,26,27,28},
                                {31,32,33,34,35,36,37,38},
                                {41,42,43,44,45,46,47,48},
                                {51,52,53,54,55,56,57,58},
                                {61,62,63,64,65,66,67,68},
                                {71,72,73,74,75,76,77,78},
                                {81,82,83,84,85,86,87,88},
                                {91,92,93,94,95,96,97,98},
                                {101,102,103,104,105,106,106,108}};

unsigned char default_H2fuelmap_duration[nthrottle][nspeed] =
                               {{5,25,10,10,10,10,10,6},
                                {10,10,10,10,10,10,10,10},
                                {10,10,25,25,25,25,11,25},
                                {21,25,25,25,25,12,25,25},
                                {25,22,25,25,13,25,25,25},
                                {25,25,23,14,25,25,25,25},
                                {25,25,15,24,25,25,25,25},
                                {25,16,25,25,25,25,25,25},
                                {17,25,25,25,25,26,25,25},
                                {25,25,25,25,25,25,27,25},
                                {8,25,25,25,25,25,25,7}};

unsigned char H2fuelmap_start[nthrottle][nspeed];
unsigned char H2fuelmap_duration[nthrottle][nspeed];

unsigned char Petrolfuelmap_start[nthrottle][nspeed];
unsigned char Petrolfuelmap_duration[nthrottle][nspeed];

unsigned int m_ThrottlePosition;
unsigned int m_Speed;

unsigned int m_CurrentColumn;
unsigned int m_CurrentRow;

unsigned int m_petroldurationangle;



//---------- PREDICTION FUNCTIONS ----------------------------------------

float Normalize(float value,float min,float max)
{
  return (value-min)/(max-min);
}
//------------------------------------------------------------------------------
float Denormalize(float normvalue,float min,float max)
{
  return (normvalue*(max-min))+min;
}
//------------------------------------------------------------------------------
void SetFirstInputElement(float *ptrinputs)
{
  *ptrinputs = 1;
}
//------------------------------------------------------------------------------
void SetFirstHiddenOutputElement(float *ptrhiddenoutputs)
```

```
{
  *ptrhiddenoutputs = 1;
}
//---------------------------------------------------------------------------
void ReadInAndHiddenWeights(int numberinputs,int numberoutputs,int
numberhiddennodes,char *filename,float *pinputweights,float *phiddenweights)
{
  FILE *fIn;
  int i,j;
  float wIn;
  float wHidden;
  fIn=fopen(filename,"rt");
  for (j=0;j<numberhiddennodes;j++)
  {
    for (i=0;i<=numberinputs;i++)
    {
      fscanf(fIn,"%f",&wIn);
      *(pinputweights+j*(numberinputs+1)+i) = wIn;
    }
    fscanf(fIn,"\n");
  }
  fscanf(fIn,"\n");
  for (i=0;i<=numberhiddennodes;i++)
  {
    for (j=0;j<numberoutputs;j++)
    {
      fscanf(fIn,"%f",&wHidden);
      *(phiddenweights+j*(numberhiddennodes+1)+i) = wHidden;
    }
    fscanf(fIn,"\n");
  }
  fclose(fIn);
}
//---------------------------------------------------------------------------
void ReadWeights()
{

ReadInAndHiddenWeights(nInputs,nOutputs,nHiddenNeuronsCO,COweightfilename,pt
rCOinWeights,ptrCOhiddenWeights);

ReadInAndHiddenWeights(nInputs,nOutputs,nHiddenNeuronsHC,HCweightfilename,pt
rHCinWeights,ptrHChiddenWeights);

ReadInAndHiddenWeights(nInputs,nOutputs,nHiddenNeuronsNO,NOweightfilename,pt
rNOinWeights,ptrNOhiddenWeights);
}
//---------------------------------------------------------------------------
void PrintWeights(int ninputs,int noutputs,int nhiddenneurons,float
*ptrinweights,float *ptrhiddenweights)
{
  int i,j;
  for (j=0;j<nhiddenneurons;j++)
  {
    for (i=0;i<=ninputs;i++)
    {
      printf("%.5f  ",(*(ptrinweights+j*(ninputs+1)+i)));
    }
```

```
    printf("\n");
  }
  printf("\n");
  for (j=0;j<noutputs;j++)
  {
    for (i=0;i<=nhiddenneurons;i++)
    {
      printf("%.5f    ",*(ptrhiddenweights+j*(nhiddenneurons+1)+i));
    }
    printf("\n");
  }
  printf("\n");
}
//---------------------------------------------------------------------------
--
void InitializePtr()
{
  ptrInputs    = (float *)malloc(sizeof(float)*(nInputs+1));
  ptrCOoutput = (float *)malloc(sizeof(float)*nOutputs);
  ptrHCoutput = (float *)malloc(sizeof(float)*nOutputs);
  ptrNOoutput = (float *)malloc(sizeof(float)*nOutputs);
  ptrHiddenOutputs    = (float *)malloc(sizeof(float)*(nHiddenNeurons+1));
  ptrCOinWeights      = (float
*)malloc(sizeof(float)*(nInputs+1)*nHiddenNeurons);
  ptrCOhiddenWeights = (float
*)malloc(sizeof(float)*(nHiddenNeurons+1)*nOutputs);
  ptrHCinWeights      = (float
*)malloc(sizeof(float)*(nInputs+1)*nHiddenNeurons);
  ptrHChiddenWeights = (float
*)malloc(sizeof(float)*(nHiddenNeurons+1)*nOutputs);
  ptrNOinWeights      = (float
*)malloc(sizeof(float)*(nInputs+1)*nHiddenNeurons);
  ptrNOhiddenWeights = (float
*)malloc(sizeof(float)*(nHiddenNeurons+1)*nOutputs);
}
//---------------------------------------------------------------------------
--
void FreePtrs()
{
  free(ptrInputs);
  free(ptrCOoutput);
  free(ptrHCoutput);
  free(ptrNOoutput);
  free(ptrHiddenOutputs);
  free(ptrCOinWeights);
  free(ptrCOhiddenWeights);
  free(ptrHCinWeights);
  free(ptrHChiddenWeights);
  free(ptrNOinWeights);
  free(ptrNOhiddenWeights);
  ptrInputs=NULL;
  ptrCOoutput=NULL;
  ptrHCoutput=NULL;
  ptrNOoutput=NULL;
  ptrHiddenOutputs=NULL;
  ptrCOinWeights=NULL;
  ptrCOhiddenWeights=NULL;
```

```
  ptrHCinWeights=NULL;
  ptrHChiddenWeights=NULL;
  ptrNOinWeights=NULL;
  ptrNOhiddenWeights=NULL;
}
//---------------------------------------------------------------------------
--
void InitializeInputVector(float *ptrinputs) //not used
{
  SetFirstInputElement(ptrinputs);
  *(ptrinputs+1) = Normalize(m_Speed,LOWER_RPM,UPPER_RPM);
  *(ptrinputs+2) =
Normalize(m_throttlevolt,LOWER_THROTTLE_V,UPPER_THROTTLE_V);
  *(ptrinputs+3) = Normalize(m_airtempvolt,LOWER_AIRTEMP_V,UPPER_AIRTEMP_V);
  *(ptrinputs+4) = Normalize(m_engtempvolt,LOWER_ENGTEMP_V,UPPER_ENGTEMP_V);
  *(ptrinputs+5) = Normalize(m_airmassvolt,LOWER_AIRMASS_V,UPPER_AIRMASS_V);
  *(ptrinputs+6) =
Normalize(H2fuelmap_start[m_CurrentRow][m_CurrentColumn],LOWER_H2_START,UPPE
R_H2_START);
  *(ptrinputs+7) =
Normalize(H2fuelmap_duration[m_CurrentRow][m_CurrentColumn],LOWER_H2_DURATIO
N,UPPER_H2_DURATION);
  *(ptrinputs+8) =
Normalize(Petrolfuelmap_start[m_CurrentRow][m_CurrentColumn],LOWER_PETROL_ST
ART,UPPER_PETROL_START);
  *(ptrinputs+9) =
Normalize(m_petroldurationangle,LOWER_PETROL_DURATION,UPPER_PETROL_DURATION)
;

}
//---------------------------------------------------------------------------
--
void CalculateOutputVector(int ninputs,int noutputs,int nhiddenneurons,float
*ptrinputs,float *ptrhiddenoutputs,float *ptroutputs,float
*ptrinweights,float *ptrhiddenweights)
{
  int i,j;
  double net;
  SetFirstHiddenOutputElement(ptrhiddenoutputs);
  //calculate HiddenOutputs
  for (j=0;j<nhiddenneurons;j++)
  {
    net = 0;
    for (i=0;i<=ninputs;i++)
    {
      net+=*(ptrinweights+j*(ninputs+1)+i)*(*(ptrinputs+i));
    }
    *(ptrhiddenoutputs+j+1)=1/(1+exp(-net));
  }
  //Calculate Outputs
  for (j=0;j<noutputs;j++)
  {
    net = 0;
    for (i=0;i<=nhiddenneurons;i++)
    {
      net +=
*(ptrhiddenweights+j*(nhiddenneurons+1)+i)*(*(ptrhiddenoutputs+i));
```

```
    }
    *(ptroutputs+j) = net;
  }
}

//----------- CONTROL FUNCTIONS ------------------------------------------

//------------------------------------------------------------------------
int InitPROMBoard(void)
{
  if (dscInit(DSC_VERSION)!=DE_NONE )
  {
    dscGetLastError(&errorParams);
    fprintf(stderr,"%s %s\n",ERROR_PREFIX,errorParams.errstring);
    return 0;
  }
  dsccb.boardtype = DSC_PROM;
  dsccb.base_address = 0x280;
  dsccb.int_level = 5;
  if(dscInitBoard(DSC_PROM, &dsccb, &dscb)!= DE_NONE)
  {
    dscGetLastError(&errorParams);
    fprintf(stderr, "%s %s\n", ERROR_PREFIX, errorParams.errstring);
    return 0;
  }
  dscadsettings.range = 1;
  dscadsettings.polarity = 1;
  dscadsettings.gain = 2;
  dscadsettings.load_cal = 0;
  dscadsettings.current_channel = 0;
  if ((result=dscADSetSettings(dscb,&dscadsettings))!=DE_NONE)
  {
    fprintf(stderr,"%s%s\n",ERROR_PREFIX,dscGetErrorString(result));
    return 0;
  }
  dscadscan.low_channel = 0;
  dscadscan.high_channel = 3;
  dscadscan.gain = dscadsettings.gain;
  dscadscan.sample_values = (DSCSAMPLE*)malloc( sizeof(DSCSAMPLE) * (
dscadscan.high_channel - dscadscan.low_channel + 1 ) );
  pfunc = CAMInterruptHandler;    //Handler to the CAM interrupt
  dscuserint.intsource = 1;         //External CAM trigger
  if
((result=dscUserInt(dscb,&dscuserint,(DSCUserInterruptFunction)pfunc))!=DE_N
ONE)
  {
    dscGetLastError(&errorParams);
    fprintf(stderr,"%s %s \n",ERROR_PREFIX,errorParams.errstring);
    return 0;
  }
  outp(BASE+4,inp(BASE+4)|0x40);  //Select 100KHz counter
  outp(BASE+15,0x88);             //Stop the counter
  outp(BASE+15,0xA0);             //Disable the counter gate
  outp(BASE+15,0x81);             //Clear the counter

  config_byte = 0x80;             //I\O port config - A,B,C are all ouput
ports
```

```
  if ((result=dscDIOSetConfig(dscb,&config_byte))!=DE_NONE) //set DIO
configuration
  {
    dscGetLastError(&errorParams);
    fprintf( stderr, "%s %s\n", ERROR_PREFIX, errorParams.errstring );
    return 0;
  }

  OFF_device(DEVICE0);
  OFF_device(DEVICE1);
  OFF_device(DEVICE2);
  OFF_device(DEVICE3);
  outportb(CONTROL,inportb(CONTROL)&0xFE);     //Initialize STROBE = 1
  return 1;
}
//-------------------------------------------------------------------------
--
int ON_device(BYTE ID)
{
  if (dscDIOSetBit(dscb,1,ID)!=DE_NONE)
  {
    dscGetLastError(&errorParams);
    fprintf( stderr, "%s %s\n", ERROR_PREFIX, errorParams.errstring );
    return 0;
  }
  return 1;
}
//-------------------------------------------------------------------------
--
int OFF_device(BYTE ID)
{
  if (dscDIOClearBit(dscb,1,ID)!=DE_NONE)
  {
    dscGetLastError(&errorParams);
    fprintf( stderr, "%s %s\n", ERROR_PREFIX, errorParams.errstring );
    return 0;
  }
  return 1;
}
//-------------------------------------------------------------------------
--
int SetHC12BoardState(int state)  //state = HC12RUN or HC12STOP
{
  if ((result=dscDIOOutputByte(dscb,0,state))!=DE_NONE)
  {
    dscGetLastError(&errorParams);
    fprintf(stderr,"%s %s\n", ERROR_PREFIX, errorParams.errstring );
    return 0;
  }
  return 1;
}
//-------------------------------------------------------------------------
--
void CAMInterruptHandler()
{
  outp(BASE+15,0xC0);   //Latch the counter
```

```
  LSB = inp(BASE+12);    //Read LSB of the counter
  MSB = inp(BASE+13);    //Read MSB of the counter
  outp(BASE+12,0xFF);    //LSB=FF to load into the counter
  outp(BASE+13,0xFF);    //MSB=FF to load into the counter
  outp(BASE+15,0x82);    //Load into the counter
  outp(BASE+15,0x84);    //Enable the counter, count down operation
}
//---------------------------------------------------------------------------
--
void SampleAnalogSignal(FLOAT &airmassvolt,FLOAT &airtempvolt,FLOAT
&engtempvolt,FLOAT &throttlevolt)
{
  /* Air Mass (V)
     Air Temperature (V)
     Engine Temperature (V)
     Throttle Position (V)
  */
  if( (result = dscADScan( dscb, &dscadscan, dscadscan.sample_values )) !=
DE_NONE )
  {
      fprintf( stderr, "%s%s\n", ERROR_PREFIX, dscGetErrorString( result )
);
      free( samples ); // remember to deallocate malloc() memory
      return 0;
  }
  /*
  for (i = 0; i < dscadscan.high_channel- dscadscan.low_channel + 1; i++)
  {
    gotoxy(25,i+2);
    voltage = ((short)dscadscan.sample_values[i] + 32768L ) *
(5.0f/65536.0f) ;
    printf("%f",voltage);
  }
  */
  airmassvolt  = ((short)dscadscan.sample_values[0] + 32768L ) *
(5.0f/65536.0f);
  airtempvolt  = ((short)dscadscan.sample_values[1] + 32768L ) *
(5.0f/65536.0f);
  engtempvolt  = ((short)dscadscan.sample_values[2] + 32768L ) *
(5.0f/65536.0f);
  throttlevolt = ((short)dscadscan.sample_values[3] + 32768L ) *
(5.0f/65536.0f);
}
//---------------------------------------------------------------------------
--
unsigned int EngineSpeedCalculation(void)
{
  unsigned int counter_data=0,speed=0;
  counter_data = MSB*256+LSB;
  speed = 120*100000/(65535-counter_data); //Nrpm=2*60*fcam=2*60/Tcam
  return speed;
}
//---------------------------------------------------------------------------
--
unsigned int ThrottlePositionCalculation(void)
{
```

```
  return (unsigned int)(m_throttlevolt-
THROTTLE_VOLT_MIN)*100/THROTTLE_VOLT_INTERVAL;
}
//---------------------------------------------------------------------------
--
int Initialization(void)
{
  MonitorSetup();
  return InitPROMBoard();
}
//---------------------------------------------------------------------------
-
void Dispose(void)
{
  dscFree();
}
//---------------------------------------------------------------------------
-
void LoadDefaultH2FuelMaps()
{
  for (int j=0;j<nthrottle;j++)
  {
    for (int i=0;i<nspeed;i++)
    {
      H2fuelmap_start[j][i]=default_H2fuelmap_start[j][i];
      H2fuelmap_duration[j][i]=default_H2fuelmap_duration[j][i];
    }
  }
}
//---------------------------------------------------------------------------
-
void LoadH2FuelMapsFromFile()
{
  ReadDataFromFile(H2filename,H2fuelmap_start,H2fuelmap_duration);
}
//---------------------------------------------------------------------------
-
void SaveH2FuelMapsToFile(void)
{
  WriteDataToFile(H2filename,H2fuelmap_start,H2fuelmap_duration);
}
//---------------------------------------------------------------------------
void LoadPetrolFuelMapsFromFile()
{

ReadDataFromFile(Petrolfilename,Petrolfuelmap_start,Petrolfuelmap_duration);
}
//---------------------------------------------------------------------------
-
void SavePetrolFuelMapsToFile(void)
{

WriteDataToFile(Petrolfilename,Petrolfuelmap_start,Petrolfuelmap_duration);
}
//---------------------------------------------------------------------------
void MixingExperiment()
{
```

```
  int HC12state=0;
  gotoxy(1,1);
  printf("Petrol-Hydrogen mixing Experiment");
  gotoxy(1,2);
  printf("(Press ESC to exit)");
  gotoxy(1,3);
  printf("Throttle position (%): ");
  gotoxy(1,4);
  printf("Engine speed (RPM): ");
  gotoxy(1,5);
  printf("Air temperature (V): ");
  gotoxy(1,6);
  printf("Engine temperature (V): ");
  gotoxy(1,7);
  printf("Air mass flow (V): ");
  gotoxy(1,8);
  printf("--Petrol--");
  gotoxy(1,9);
  printf("SOI (deg.): ");
  gotoxy(1,10);
  printf("DOI (deg.): ");
  gotoxy(1,11);
  printf("--Hydrogen--");
  gotoxy(1,12);
  printf("SOI (deg.): ");
  gotoxy(1,13);
  printf("DOI (deg.): ");
  gotoxy(1,14);
  printf("--Emissions Prediction--");
  gotoxy(1,15);
  printf("CO   (%): ");
  gotoxy(1,16);
  printf("HC (ppm): ");
  gotoxy(1,17);
  printf("NO (ppm): ");

  //gotoxy(1,18);
  //printf("HC12 board state                    =");
  while (!kbhit())
  {
    ClearWindow(35,2,50,12);

SampleAnalogSignal(m_airmassvolt,m_airtempvolt,m_engtempvolt,m_throttlevolt)
;
    m_Speed=EngineSpeedCalculation();
    m_ThrottlePosition=ThrottlePositionCalculation();
    if (CheckValueInRange(m_Speed,array_speed,nspeed)==1 &&
CheckValueInRange(m_ThrottlePosition,array_throttle,nthrottle)==1)
    {
      m_CurrentRow =
ValueIndex(m_ThrottlePosition,array_throttle,nthrottle);
      m_CurrentColumn = ValueIndex(m_Speed,array_speed,nspeed);
      SetHC12BoardState(HC12RUN);
      HC12state = 1;

SendInjectionData(H2fuelmap_start[m_CurrentRow][m_CurrentColumn],H2fuelmap_d
uration[m_CurrentRow][m_CurrentColumn]);
```

```
    }
    else
    {
      m_CurrentRow = 0;
      m_CurrentColumn = 0;
      SetHC12BoardState(HC12STOP);
      HC12state = 0;
    }
    //gotoxy(37,3);
    //printf("%f",m_throttlevolt);//Throttle Position (V)
    gotoxy(37,3);
    printf("%d",m_ThrottlePosition);//Throttle position(%)
    gotoxy(37,4);
    printf("%d",m_Speed);//Engine speed(RPM)
    gotoxy(37,5);
    printf("%f",m_airtempvolt);//Air Temperature (V)
    gotoxy(37,6);
    printf("%f",m_engtempvolt);//Engine Temperature (V)
    gotoxy(37,7);
    printf("%f",m_airmassvolt);// Air Mass Flow (V)
    gotoxy(37,9)
    printf("%d",Petrolfuelmap_start[m_CurrentRow][m_CurrentColumn]);//Start
of Petrol Injection(deg)
    gotoxy(37,10);
    m_petroldurationangle =
Petrolfuelmap_duration[m_CurrentRow][m_CurrentColumn]*m_Speed*6/1000;
    printf("%d",m_petroldurationangle);//Duration of Petrol Injection(deg)
=");
    gotoxy(37,12);
    printf("%d",H2fuelmap_start[m_CurrentRow][m_CurrentColumn]);//Start of
H2 Injection(deg)
    gotoxy(37,13);

printf("%d",H2fuelmap_duration[m_CurrentRow][m_CurrentColumn]);//Duration of
H2 Injection(deg)
    InitializeInputVector(ptrInputs);

CalculateOutputVector(nInputs,nOutputs,nHiddenNeuronsCO,ptrInputs,ptrHiddenO
utputs,ptrCOoutput,ptrCOinWeights,ptrCOhiddenWeights)

CalculateOutputVector(nInputs,nOutputs,nHiddenNeuronsHC,ptrInputs,ptrHiddenO
utputs,ptrHCoutput,ptrHCinWeights,ptrHChiddenWeights)

CalculateOutputVector(nInputs,nOutputs,nHiddenNeuronsNO,ptrInputs,ptrHiddenO
utputs,ptrNOoutput,ptrNOinWeights,ptrNOhiddenWeights)
    gotoxy(37,15);
    printf("%f",*ptrCOoutput);//CO prediction
    gotoxy(37,16);
    printf("%f",*ptrHCoutput);//HC prediction
    gotoxy(37,17);
    printf("%f",*ptrNOoutput);//NO prediction

    //printf("HC12 board state                    =");
    //printf("%d",HC12state);
  }
  SetHC12BoardState(HC12STOP);
}
```

```
//-------------------------------------------------------------------------
void StartInjectionTuning()
{
  char ch=0;
  char *str="";
  char *digitstr;
  int number;
  window(1,1,80,25);
  clrscr();
  *str='\0';
  WriteStringXY(0,0,"Start of Injection Tuning");
  WriteStringXY(0,1,"(Press ESC to exit)");
  WriteStringXY(0,2,"Throttle position(%): ");
  WriteStringXY(0,3,"Engine speed(RPM): ");
  WriteStringXY(0,4,"Start of Injection Angle(deg.):");
  WriteStringXY(0,5,"Duration of Injection Angle(deg.):");
  WriteStringXY(0,6,"Enter new Start of Injection Angle(deg.):");
  while (ch!=27)
  {
    WriteStringXY(0,0,"Start of Injection Tuning");
    //m_Speed=random(3500)+1500;
    //m_ThrottlePosition=random(100);

SampleAnalogSignal(m_airmassvolt,m_airtempvolt,m_engtempvolt,m_throttlevolt)
;
    m_Speed=EngineSpeedCalculation();
    m_ThrottlePosition=ThrottlePositionCalculation();
    if (CheckValueInRange(m_Speed,array_speed,nspeed)==1 &&
CheckValueInRange(m_ThrottlePosition,array_throttle,nthrottle)==1)
    {
      m_CurrentRow =
ValueIndex(m_ThrottlePosition,array_throttle,nthrottle);
      m_CurrentColumn = ValueIndex(m_Speed,array_speed,nspeed);
      SetHC12BoardState(HC12RUN);

SendInjectionData(H2fuelmap_start[m_CurrentRow][m_CurrentColumn],H2fuelmap_d
uration[m_CurrentRow][m_CurrentColumn]);
    }
    else
    {
      m_CurrentRow = 0;
      m_CurrentColumn = 0;
      SetHC12BoardState(HC12STOP);
    }
    ClearWindow(22,2,27,2);
    WriteIntXY(22,2,array_throttle[m_CurrentRow]);
    ClearWindow(19,3,25,3);
    WriteIntXY(19,3,array_speed[m_CurrentColumn]);
    ClearWindow(32,4,38,4);
    WriteIntXY(32,4,H2fuelmap_start[m_CurrentRow][m_CurrentColumn]);
    ClearWindow(35,5,41,5);
    WriteIntXY(35,5,H2fuelmap_duration[m_CurrentRow][m_CurrentColumn]);
    if (KeyReady())
    {
      ch=GetKeyNumber();
      if (ch==27)
      {
```

```
      return;
      }
      if (ch>=48&&ch<=57)
      {
      if (strlen(str)==3)
      {
        *str='\0';
      }
      else
      {
        *digitstr=ch;
        *(digitstr+1)='\0';
        strcat(str,digitstr);
      }
        ClearWindow(42,6,50,6);
        WriteStringXY(42,6,str);
      }
      if (ch==13)
      {
      number=atoi(str);
      if ((number<180)&&(number>=1))
      {
        H2fuelmap_start[m_CurrentRow][m_CurrentColumn]=number;
      }
      }
    }
  }
}
//-------------------------------------------------------------------------
void DurationInjectionTuning()
{
  char ch=0;
  char *str="";
  char *digitstr;
  int number;
  window(1,1,80,25);
  clrscr();
  *str='\0';
  WriteStringXY(0,0,"Duration of Injection Tuning");
  WriteStringXY(0,1,"(Press ESC to exit)");
  WriteStringXY(0,2,"Throttle position(%): ");
  WriteStringXY(0,3,"Engine speed(RPM): ");
  WriteStringXY(0,4,"Start of Injection Angle(deg.):");
  WriteStringXY(0,5,"Duration of Injection Angle(deg.):");
  WriteStringXY(0,6,"Enter new Start of Injection Angle(deg.):");
  while (ch!=27)
  {
    WriteStringXY(0,0,"Duration of Injection Tuning");
    //m_Speed=random(3500)+1500;
    //m_ThrottlePosition=random(100);

SampleAnalogSignal(m_airmassvolt,m_airtempvolt,m_engtempvolt,m_throttlevolt)
;
    m_Speed=EngineSpeedCalculation();
    m_ThrottlePosition=ThrottlePositionCalculation();
    if (CheckValueInRange(m_Speed,array_speed,nspeed)==1 &&
CheckValueInRange(m_ThrottlePosition,array_throttle,nthrottle)==1)
```

```
    {
      m_CurrentRow =
ValueIndex(m_ThrottlePosition,array_throttle,nthrottle);
      m_CurrentColumn = ValueIndex(m_Speed,array_speed,nspeed);
      SetHC12BoardState(HC12RUN);

SendInjectionData(H2fuelmap_start[m_CurrentRow][m_CurrentColumn],H2fuelmap_d
uration[m_CurrentRow][m_CurrentColumn]);
    }
    else
    {
      m_CurrentRow = 0;
      m_CurrentColumn = 0;
      SetHC12BoardState(HC12STOP);
    }
    ClearWindow(22,2,27,2);
    WriteIntXY(22,2,array_throttle[m_CurrentRow]);
    ClearWindow(19,3,25,3);
    WriteIntXY(19,3,array_speed[m_CurrentColumn]);
    ClearWindow(32,4,38,4);
    WriteIntXY(32,4,H2fuelmap_start[m_CurrentRow][m_CurrentColumn]);
    ClearWindow(35,5,41,5);
    WriteIntXY(35,5,H2fuelmap_duration[m_CurrentRow][m_CurrentColumn]);
    //delay(100);
    if (KeyReady())
    {
      ch=GetKeyNumber();
      if (ch==27)
      {
      return;
      }
      if (ch>=48&&ch<=57)
      {
      if (strlen(str)==3)
      {
        *str='\0';
      }
      else
      {
        *digitstr=ch;
        *(digitstr+1)='\0';
        strcat(str,digitstr);
      }
      ClearWindow(42,6,50,6);
      WriteStringXY(42,6,str);
      }
      if (ch==13)
      {
      number=atoi(str);
      if ((number<180)&&(number>=1))
      {
        H2fuelmap_duration[m_CurrentRow][m_CurrentColumn]=number;
      }
      }
    }
  }
}
```

```
//-------------------------------------------------------------------------
void ViewStartInjectionMap(unsigned char x,unsigned char y,unsigned char
map_start[nthrottle][nspeed])
{
  int i,j;
  int k=0,l=0;
  char ch=0;
  char *str="";
  char *digitstr;
  int number;

START_MAP:
  *str='\0';
  ClearWindow(0,0,79,24);
  WriteStringXY(0,0,"*");
  WriteStringXY(x+14,y+2,"Speed");
  WriteStringXY(x,y+5,"Throttle");
  for (i=0;i<nspeed;i++)
  {
    WriteIntXY(x+14+6*i,y+4,array_speed[i]);
  }
  for (j=0;j<nthrottle;j++)
  {
    WriteIntXY(x+9,y+5+j,array_throttle[j]);
  }
  for (j=0;j<nthrottle;j++)
    for (i=0;i<nspeed;i++)
      WriteIntXY(x+14+6*i,y+5+j,map_start[j][i]);
  WriteStringXY(x,y+7+nthrottle,"Speed:");
  WriteIntXY(x+7,y+7+nthrottle,array_speed[k]);
  WriteStringXY(x,y+8+nthrottle,"Throttle:");
  WriteIntXY(x+10,y+8+nthrottle,array_throttle[l]);
  WriteStringXY(x,y+9+nthrottle,"Current value:");
  WriteIntXY(x+15,y+9+nthrottle,map_start[l][k]);
  WriteStringXY(x,y+10+nthrottle,"New value:");
  ClearWindow(0,0,5,0);
  WriteStringXY(x+17,y,"Start of Injection Map (deg.)");

  while (ch!=27)
  {
    if (KeyReady())
    {
      ch=GetKeyNumber();
      if
(ch==UP||ch==DOWN||ch==LEFT||ch==RIGHT||ch==up||ch==down||ch==left||ch==righ
t)
      {
      if (ch==UP||ch==up)
      {
        if (l==0)
          l=nthrottle-1;
        else
          l--;
      }
      if (ch==DOWN||ch==down)
      {
        if (l==nthrottle-1)
```

```
            l=0;
          else
            l++;
        }
        if (ch==LEFT||ch==left)
        {
          if (k==0)
            k=nspeed-1;
          else
            k--;
        }
        if (ch==RIGHT||ch==right)
        {
          if (k==nspeed-1)
            k=0;
          else
            k++;
        }
        goto START_MAP;
        }
        if (ch>=48&&ch<=57)
        {
        if (strlen(str)==3)
        {
          *str='\0';
        }
        else
        {
          *digitstr=ch;
          *(digitstr+1)='\0';
          strcat(str,digitstr);
        }
        ClearWindow(x+11,y+10+nthrottle,x+11+8,y+10+nthrottle);
        WriteStringXY(x+11,y+10+nthrottle,str);
        }
        if (ch==13)
        {
        number=atoi(str);
        if ((number<180)&&(number>=1))
        {
          map_start[l][k]=number;
          goto START_MAP;
        }
        }
      }
    }
}
//-------------------------------------------------------------------------
void ViewDurationInjectionMap(unsigned char x,unsigned char y,unsigned char
map_duration[nthrottle][nspeed])
{
  int i,j;
  int k=0,l=0;
  char ch=0;
  char *str="";
  char *digitstr;
  int number;
```

```
DURATION_MAP:
  *str='\0';
  ClearWindow(0,0,79,24);
  WriteStringXY(0,0,"*");
  WriteStringXY(x+14,y+2,"Speed");
  WriteStringXY(x,y+5,"Throttle");
  for (i=0;i<nspeed;i++)
  {
    WriteIntXY(x+14+6*i,y+4,array_speed[i]);
  }
  for (j=0;j<nthrottle;j++)
  {
    WriteIntXY(x+9,y+5+j,array_throttle[j]);
  }
  for (j=0;j<nthrottle;j++)
    for (i=0;i<nspeed;i++)
      WriteIntXY(x+14+6*i,y+5+j,map_duration[j][i]);
  WriteStringXY(x,y+7+nthrottle,"Speed:");
  WriteIntXY(x+7,y+7+nthrottle,array_speed[k]);
  WriteStringXY(x,y+8+nthrottle,"Throttle:");
  WriteIntXY(x+10,y+8+nthrottle,array_throttle[l]);
  WriteStringXY(x,y+9+nthrottle,"Current value:");
  WriteIntXY(x+15,y+9+nthrottle,map_duration[l][k]);
  WriteStringXY(x,y+10+nthrottle,"New value:");
  ClearWindow(0,0,5,0);
  WriteStringXY(x+17,y,"Duration of Injection Map (deg.)");

  while (ch!=27)
  {
    if (KeyReady())
    {
      ch=GetKeyNumber();
      if
(ch==UP||ch==DOWN||ch==LEFT||ch==RIGHT||ch==up||ch==down||ch==left||ch==righ
t)
      {
      if (ch==UP||ch==up)
      {
        if (l==0)
          l=nthrottle-1;
        else
          l--;
      }
      if (ch==DOWN||ch==down)
      {
        if (l==nthrottle-1)
          l=0;
        else
          l++;
      }
      if (ch==LEFT||ch==left)
      {
        if (k==0)
          k=nspeed-1;
        else
          k--;
```

```
      }
      if (ch==RIGHT||ch==right)
      {
        if (k==nspeed-1)
          k=0;
        else
          k++;
      }
      goto  DURATION_MAP;
      }
      if (ch>=48&&ch<=57)
      {
      if (strlen(str)==3)
      {
        *str='\0';
      }
      else
      {
        *digitstr=ch;
        *(digitstr+1)='\0';
        strcat(str,digitstr);
      }
      ClearWindow(x+11,y+10+nthrottle,x+11+8,y+10+nthrottle);
      WriteStringXY(x+11,y+10+nthrottle,str);
      }
      if (ch==13)
      {
      number=atoi(str);
      if ((number<180)&&(number>=1))
      {
        map_duration[l][k]=number;
        goto DURATION_MAP;
      }
      }
    }
  }
}
//-------------------------------------------------------------------------
void ViewStartInjectionH2Map(unsigned char x,unsigned char y)
{
  ViewStartInjectionMap(x,y,H2fuelmap_start[nthrottle][nspeed]);
}
//-------------------------------------------------------------------------
void ViewDurationInjectionH2Map(unsigned char x,unsigned char y)
{
  ViewDurationInjectionMap(x,y,H2fuelmap_duration[nthrottle][nspeed]);
}
//-------------------------------------------------------------------------
void ViewStartInjectionPetrolMap(unsigned char x,unsigned char y)
{
  ViewStartInjectionMap(x,y,Petrofuelmap_start[nthrottle][nspeed]);
}
//-------------------------------------------------------------------------
void ViewDurationInjectionPetrolMap(unsigned char x,unsigned char y)
{
  ViewDurationInjectionMap(x,y,Petrolfuelmap_duration[nthrottle][nspeed]);
}
```

```
//-------------------------------------------------------------------------
void PrintMainMenu(void)
{
  char ch=0;
  InitializePtr();
  ReadWeights()
  LoadH2FuelMapsFromFile();
  LoadPetrolFuelMapsFromFile();
  while ((ch!='V')&&(ch!='v'))
  {
    ch=0;
    clrscr();
    SetHC12BoardState(HC12STOP);
    outportb(CONTROL,inportb(CONTROL)&0xFE);     //Initialize STROBE = 1
    WriteStringXY(0,0,"H2 Injection System with Virtual Gas Sensor");
    WriteStringXY(0,2,"1 - Petrol - H2 Mixing experiment (ONLY ACTIVE AFTER
ENGINE HAS BEEN CRANKED)");
    WriteStringXY(0,3,"2 - Start of Injection of H2 tuning");
    WriteStringXY(0,4,"3 - Duration of Injection of H2 tuning");
    WriteStringXY(0,5,"4 - View start of H2 injection map");
    WriteStringXY(0,6,"5 - View duration of H2 injection map");
    WriteStringXY(0,7,"6 - Load default H2 fuel maps");
    WriteStringXY(0,8,"7 - Load H2 fuel maps from file");
    WriteStringXY(0,9,"8 - Save H2 fuel maps to file");
    WriteStringXY(0,10,"9 - View start of Petrol injection map");
    WriteStringXY(0,11,"Z - View duration of Petrol injection map");
    WriteStringXY(0,12,"X - Load Petrol fuel maps from file");
    WriteStringXY(0,13,"C - Save Petrol fuel maps to file");
    WriteStringXY(0,14,"V - Exit");
    ch = getch();
    switch (ch)
    {
      case '1':MixingExperiment();
               break;
      case '2':StartInjectionTuning();
               break;
      case '3':DurationInjectionTuning();
               break;
      case '4':ViewStartInjectionH2Map(6,2);
               break;
      case '5':ViewDurationInjectionH2Map(6,2);
               break;
      case '6':LoadDefaultH2FuelMaps();
               break;
      case '7':LoadH2FuelMapsFromFile();
               break;
      case '8':SaveH2FuelMapsToFile();
               break;
      case '9':ViewStartInjectionPetrolMap(6,2);
               break;
      case 'Z':ViewDurationInjectionPetrolMap(6,2);
               break;
      case 'z':ViewDurationInjectionPetrolMap(6,2);
               break;
      case 'X':LoadPetrolFuelMapsFromFile();
               break;
      case 'x':LoadPetrolFuelMapsFromFile();
```

```
                break;
        case 'C':SavePetrolFuelMapsToFile();
                break;
        case 'c':SavePetrolFuelMapsToFile();
                break;
        case 'V':Dispose();
                return;
        case 'v':Dispose();
                return;
      }
    }
}
//---------------------------------------------------------------------------
void main(void)
{
  if (Initialization()==0)
  {
    return;
  }
  PrintMainMenu();
}
```

Supporting file "inj2.cpp"

```
#include <conio.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include "inj.h"
//#include "dscud.h"

//---------------------------------------------------------------------------
-
void MonitorSetup(void)
{
  clrscr();
  _AH = 0x00;
  _AL = 0x03+0x80;
  _BH = 0;
  geninterrupt(0x10);
}
//---------------------------------------------------------------------------
-
int KeyReady()
{
  long int x;
  _AH=1;
  geninterrupt(0x16);
  x=_FLAGS&0x40;
  if (x==0)
  return 1;
  else
  return 0;
}
//---------------------------------------------------------------------------
char GetKeyNumber()
{
  char a;
  _AH=0x00;
  geninterrupt(0x16);
  a=_AL;
  return a;
}
//---------------------------------------------------------------------------
void WriteStringXY(unsigned char x, unsigned char y, char *st)
{
  int i=0;
  int slength = strlen(st);
  while (i<slength)
  {
    WriteCharXY(x+i,y,*(st+i));
    i++;
  }
}
```

```
//-------------------------------------------------------------------------
--
void WriteIntXY(unsigned char x, unsigned char y, int number)
{
  char *str;
  itoa(number,str,10);
  WriteStringXY(x,y,str);
}
//-------------------------------------------------------------------------
--
void WriteCharXY(unsigned char x,unsigned char y,char ch)
{
  _AH = 0x02;
  _DH = y;
  _DL = x;
  _BH = 0;
  geninterrupt(0x10);
  struct text_info ti;
  /* grab current text settings */
  gettextinfo(&ti);
  _AH = 9;
  _AL = ch;
  _BH = 0;                    /* video page */
  _BL = ti.attribute;  /* video attribute */
  _CX = 1;            /* repetition factor */
  geninterrupt(0x10);  /* output the char */
}
//-------------------------------------------------------------------------
-
void ClearWindow(unsigned char x1,unsigned char y1,unsigned char x2,unsigned
char y2)
{
  window(x1+1,y1+1,x2+1,y2+1);
  clrscr();
  window(1,1,80,25);
}
//-------------------------------------------------------------------------
-
void ConvertDimension(unsigned char OneD[nthrottle*nspeed],unsigned char
TwoD[nthrottle][nspeed])
{
  int x=0,y=0,i;
  for(i=0; i<nthrottle*nspeed; i++)
  {
    TwoD[y][x] = OneD[i];
    if ((i+1)%nspeed==0)
    {
      y++;
      x=0;
    }
    else
    {
      x++;
    }
  }
}
//-------------------------------------------------------------------------
```

```
bool WriteDataToFile(char *filename,unsigned char
map_start[nthrottle][nspeed],unsigned char map_duration[nthrottle][nspeed])
{
  FILE *fp;
  int i;
  unsigned char (*ptr)[nspeed];
  if ((fp=fopen(filename, "wb"))==NULL)
  {
    return false;
  }
  else
  {
    ptr = map_start;
    for (i=0;i<nthrottle;i++)
    {
      if(fwrite((ptr+i), sizeof(unsigned char), nspeed, fp) != nspeed)
      return false;
    }
    ptr = map_duration;
    for (i=0;i<nthrottle;i++)
    {
      if(fwrite((ptr+i), sizeof(unsigned char), nspeed, fp) != nspeed)
      return false;
    }
    fclose(fp);
    return true;
  }
}
//---------------------------------------------------------------------------
bool ReadDataFromFile(char *filename,unsigned char
map_start[nthrottle][nspeed],unsigned char map_duration[nthrottle][nspeed])
{
  FILE *fp;
  unsigned char result[nthrottle*nspeed];
  unsigned char *ptr_result=result;
  if((fp=fopen(filename, "rb"))==NULL)
  {
    return false;
  }
  else
  {
    if(fread(ptr_result, sizeof(unsigned char), nthrottle*nspeed, fp) !=
nthrottle*nspeed)
    {
      return false;
    }
    ConvertDimension(result,map_start);
    fseek(fp,nthrottle*nspeed,SEEK_SET);
    if(fread(ptr_result, sizeof(unsigned char), nthrottle*nspeed, fp) !=
nthrottle*nspeed)
    {
      return false;
    }
    ConvertDimension(result,map_duration);
    fclose(fp);
    return true;
  }
```

```
}
//-------------------------------------------------------------------------
unsigned char CheckValueInRange(unsigned int value,unsigned int
*items,unsigned int n_items)
{
  if (value>=items[0] && value<=items[n_items-1])
  return 1;
  else
  return 0;
}
//-------------------------------------------------------------------------
unsigned char ValueIndex(unsigned int value,unsigned int *items,unsigned int
n_items)
{
  unsigned int lower = 0;
  unsigned int upper = n_items-1;
  unsigned int mid;
  while (upper-lower>1)
  {
    mid = (unsigned int)((lower+upper)/2);
    if (items[mid]<=value)
    {
      lower=mid;
    }
    else
    {
      upper=mid;
    }
  }
  if ((value-items[lower])>(items[upper]-value))
  return lower+1;
  else
  return lower;
}
//-------------------------------------------------------------------------
void SendInjectionData(unsigned int inj_start_param,unsigned int
inj_dura_param)
{
  outportb(CONTROL,inportb(CONTROL)&0xFE);   //Initialize STROBE=1
  outportb(DATA,(unsigned char)inj_start_param);   //send the start of
injection
  outportb(CONTROL,inportb(CONTROL)&0xF9);   //INIT=0,AFEED=1
  delay(1);
  outportb(CONTROL,inportb(CONTROL)|0x01);   //STROBE=0
  while ((inportb(STATUS)&0x40)!=0x40)       //Check ACKNOWLEDGE
  {
  }
  outportb(CONTROL,inportb(CONTROL)&0xFE);   //reset STROBE=1
  outportb(DATA,(unsigned char)inj_dura_param);    //send the duration of
injection
  outportb(CONTROL,inportb(CONTROL)|0x06);   //INIT=1,AFEED=0
  delay(1);
  outportb(CONTROL,inportb(CONTROL)|0x01);   //STROBE=0
  while ((inportb(STATUS)&0x40)!=0x40)       //Check ACKNOWLEDGE
  {
  }
  outportb(CONTROL,inportb(CONTROL)&0xFE);   //reset STROBE=1
```

```
}
```

APPENDIX G
ACCOMPANYING COMPACT DISK

A compact disk containing neural network software, experimental data, control program (project files, source codes and resources) is provided with the thesis.